DOCUMENT RESUME

ED 359 073 SE 053 691

AUTHOR	Cornu, Bernard, Ed.; Ralston, Anthony, Ed.
TITLE	The Influence of Computers and Informatics on Mathematics and Its Teaching. Science and Technology Education Series, 44.
INSTITUTION	United Nations Educational, Scientific, and Cultural Organization, Paris (France). Div. of Science, Technical and Environmental Education.
REPORT NO	ED-92/WS/17
PUB DATE	Oct 92
NOTE	140p.
PUB TYPE	Collected Works - Serials (022)
EDRS PRICE	MF01/PC06 Plus Postage.
DESCRIPTORS	Algorithms; Calculators; Calculus; Computer Assisted Instruction; Computer Science Education; *Computer Uses in Education; *Educational Technology; Elementary Secondary Education; Foreign Countries; Inservice Teacher Education; Mathematics Curriculum; Mathematics Education; *Mathematics Instruction; Microcomputers; Preservice Teacher Education
IDENTIFIERS	Computer Algebra; Discrete Mathematics; Graphing Utilities; Representations (Mathematics)

ABSTRACT

In 1985 the International Commission on Mathematical Instruction (ICMI) published the first edition of a book of studies on the topic of the influence of computers on mathematics and the teaching of mathematics. This document is an updated version of that book and includes five articles from the 1985 ICMI conference at Strasbourg, France; reports by the leaders of three workshops held at that meeting; and four new articles on topics related to mathematics instruction using technology. The articles are: (1) "Overview" (H. Burkhardt and R. Fraser); (2)"The Effect of Computers on Mathematics" (R. F. Churchhouse); (3) "The Impact of Computers and Computer Science on the Mathematics Curriculum" (A. Ralston); (4) "Computers as an Aid to Teaching and Learning Mathematics" (B. Cornu); (5) "Living with a New Mathematical Species" (L. A. Steen); (6) "What are Algorithms? What is Algorithmics?" (S. B. Maurer); (7) "On the Mathematical Basis of Computer Science" (J. Stern); (8) "The Effect of Computers on the School Mathematics Curriculum (K.-D. Graf, R. Fraser, L. Klingen, J. Stewart, and B. Winkelmann); (9) "A Fundamental Course in Higher Mathematics Incorporating Discrete and Continuous Themes" (S. B. Seidman and M. D. Rice); (10) Teacher Education and Training" (B. Cornu); (11) "The Impact of Symbolic Mathematical Systems on Mathematics Education" (B. R. Hodgson and E. R. Muller); (12) "Calculus Teaching and the Computer. On the Interplay of Discrete Numerical Methods and Calculus in the Education of Users of Mathematics" (M. Mascarello and B. Winkelmann); and (13) "Graphic Insight into Mathematical Concepts" (D. Tall and B. West). A list of 62 annotated references and an index are included. (MDH)

ED359073

The influence of computers and informatics on mathematics and its teaching

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

F. Zanuttini

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it

☐ Minor changes have been made to improve reproduction quality

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

Science and technology education, 44

The influence of computers and informatics on mathematics and its teaching

Edited by
Professor Bernard Cornu
Professor Anthony Ralston

UNESCO

Education Sector
ED-92/WS/17

Paris
October 1992

The designations employed and the presentation of the material in this document do not imply the expression of any opinion whatsoever on the part of UNESCO concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries.

Published in 1992 by the United Nations Educational,
Scientific and Cultural Organization,
7 Place de Fontenoy,
75352 Paris 07 SP

Printed by UNESCO

© UNESCO 1992
Printed in France

Preface

The three-way interaction between mathematics, computers and mathematics education is becoming stronger each year. How schools and universities should respond is still an open question. This document has been prepared to contribute to the debate. The following quotation from the Overview chapter states succinctly why this debate is so important:

> We are facing a situation in which children are taught to do mathematics in ways that are very largely outmoded, with at least 80% of curriculum time wasted on trying, more or less successfully, to develop fluency in skills of now limited value.

The International Commission on Mathematical Instruction (ICMI) undertook a study, "The Influence of Computers and Informatics on Mathematics and Its Teaching", which included a conference in Strasbourg, France in 1985, in which UNESCO co-operated. The outcome of the Study was a book published by Cambridge University Press, bearing the conference title. With the quick pace of change of computers, mathematics and its teaching, the book's contents have become outdated. The development of this new document is explained in the Editors' Foreword.

The reader will notice that the authors of this document are all from Europe and North America. One might conclude that school uses of computers are only known in those two regions. This is certainly not true. But more abundant financial resources have permitted a greater penetration of computers in schools and universities in Europe and North America than elsewhere. For the situation in 'the rest of the world', see the reference, "An International Perspective", by Jacobsen in the Annotated References.

Unesco wishes to express its appreciation to the editors, Professors Anthony Ralston and Bernard Cornu, to the authors for their contributions, to Professor Ralston for preparing the final manuscript and to Cambridge University Press for giving its permission for UNESCO to include in this document some updated contributions from the original publication.

The views expressed in this report are those of the editors or the individual authors and not necessarily those of UNESCO.

We welcome comments on the contents of this document, which should be sent to: Mathematics Education Programme Specialist (Science and Environmental Education Section) or Mathematics and Computing Programme Specialist (Basic Sciences Division), UNESCO, Place de Fontenoy, 75700 Paris, France.

UNESCO, Paris

Editors' Foreword to Second Edition

In 1985 the International Commission on Mathematical Instruction (ICMI) chose The Influence of Computers and Informatics on Mathematics and Its Teaching as the topic of the first of a series of studies on topics of current interest within mathematics education. ICMI could not have chosen a more apt and important topic. In the seven years since the publication of the first edition of this book, the importance of calculators and computers has grown rapidly and is now the single most important factor in creating change in all aspects of mathematics education. Thus, ICMI with the cooperation of UNESCO asked us to edit an updated version of the original book which would retain the strengths of that volume but would also bring the topics in it up to date and, as well, incorporate topics which were not adequately discussed in the first edition but are now of major importance (e.g. symbolic mathematical systems, algorithms and algorithmics).

The conference at Strasbourg in 1985 whose proceedings were incorporated in the first edition was organized by a Program Committee consisting of R. F. Churchhouse (Cardiff), B. Cornu (Grenoble), A. P. Ershov (Novosibirsk), A. G. Howson (Southampton), J.-P. Kahane (Orsay), J. H. van Lint (Eindhoven), F. Pluvinage (Strasbourg), A. Ralston (Buffalo) and M. Yamaguti (Kyoto). The proceedings were edited by A. G. Howson and J.-P. Kahane and were published by Cambridge University Press.

For this edition the report of the Strasbourg meeting itself has been brought up to date by the leaders of the three workshops held at that meeting and five of the articles in the first edition have been updated for this edition. In addition, the editors have solicited four new articles written just for this edition. The result, we hope, is a volume which will be as well received as was the first edition and which will be useful to mathematics educators throughout the world.

Of course, the nature of computer and calculator technology is that it changes so rapidly - as do its implications for mathematics education - that a third edition will no doubt be needed in several years. But the current volume gives a fair picture of the impact of computers and calculators on mathematics education in 1992. It should, therefore, provide a valuable resource for mathematics educators who wish to learn about this impact or who wish to incorporate the technology into their mathematics teaching or their teaching of prospective mathematics teachers.

The editors wish to thank UNESCO for its support for this project, particularly Angelo Marzollo and Edward Jacobsen.

This book was produced using TeX at the State University of New York at Buffalo.

Bernard Cornu
Anthony Ralston

June 1992

CONTENTS

H. Burkhardt and R. Fraser: **Overview** 1

An Update of the 1985 Strasbourg conference: 11
  R. F. Churchhouse: **The Effect of Computers on Mathematics** 12
  A. Ralston: **The Impact of Computers and Computer Science on the Mathematics Curriculum** 19
  B. Cornu: **Computers as an Aid to Teaching and Learning Mathematics** 25

L. A. Steen: **Living with a New Mathematical Species** 33

S. B. Maurer: **What are Algorithms? What Is Algorithmics?** 39

J. Stern: **On the Mathematical Basis of Computer Science** 51

K.-D. Graf, R. Fraser, L. Klingen, J. Stewart and B. Winkelmann: **The Effect of Computers on the School Mathematics Curriculum** 57

S. B. Seidman and M. D. Rice: **A Fundamental Course in Higher Mathematics Incorporating Discrete and Continuous Themes** 80

B. Cornu: **Teacher Education and Training** 87

B. R. Hodgson and E. R. Muller: **The Impact of Symbolic Mathematical Systems on Mathematics Education** 93

M. Mascarello and B. Winkelmann: **Calculus Teaching and the Computer. On the Interplay of Discrete Numerical Methods and Calculus in the Education of Users of Mathematics** 108

D. Tall and B. West: **Graphic Insight into Mathematical Concepts** 117

Annotated References 124

Index 131

AN OVERVIEW

Hugh Burkhardt
University of Nottingham, U.K.

Rosemary Fraser
University of Nottingham, U.K.

The challenge

Where are we going? Where do we want to go? Why? How do we know? How may we find out more? How do we get it to happen?

As far as the influence of computers and informatics on mathematics and the mathematics curriculum is concerned, these are the central questions that this volume, like its predecessor, will address.

We shall also be concerned with progress in the seven years since the original Strasbourg meeting. Which aspects have moved quickly and substantially towards reasonably firm conclusions? On which areas is the situation now little different from then? What needs to be done about it?

A mismatch of timescales is one of the central challenges of this field which does not normally occur in the processes of change, either within mathematics itself or in the development of new curricula. The pace of change in the technology is much faster than has ever been achieved for school curricula; typical timescales for significant changes to occur are roughly as follows:

computer technology	a few years
mathematics research	10 – 20 years
school curricula	5 – 20 years

Thus we should be aware that when we design new curricula to use the power of new technology, we shall continually be behind the times. This *moving target problem* is well recognised but needs to be addressed at a strategic level in planning change. If the new curriculum elements are to be robust and widely useful, the curriculum designer cannot assume a specific level of technological provision and sophistication in schools – both will vary widely from time to time and from place to place.

This is important. If each student has a 'micro', curriculum possibilities open up which are not there with one micro per class; even these possibilities depend on the sophistication of the micro – one line of display, a few lines, many lines, graphics, access to data – each step is significant. Equally, it is already clear that even quite low levels of computer provision and sophistication still have enormous educational potential. Is 'technical restraint' a virtue, or does it impede progress?

The overall picture

It may be useful to begin with an overview of the present situation in three separate domains of activity:

A Doing Mathematics – this is the domain of mathematical activity; in every sphere, from everyday uses to research, it has been revolutionised by technology.

B Understanding of the Learning and Teaching of Mathematics – this domain is concerned with the processes of learning and teaching concepts, skills and strategies in mathematics and its applications; it is clear that technology has profound implications here, both through the changes in doing mathematics and as a potential aid to learning and teaching, but these phenomena are not yet well-understood.

C Mathematics Curricula and Teacher Training – both the first two domains have implications for curricula, including both materials and teacher support; the development of new curricula that reflect the changed learning objectives and use technology effectively in their realisation is a major task.

The pattern of change so far is summarised in Table 1 on the next page.

We shall now discuss each of the three domains in more detail.

Changes in 'doing mathematics'

In *Domain A* we now have a situation in which the changes in the way mathematics is done, at every level from the shopkeeper to the research mathematician and engineer, are moving purposefully forward with the advances in the technology, and with the methods for its utilisation that informatics helps to develop. Obviously there is some time lag but, at least in comparison with the exploitation within mathematical education, there is no serious impediment to change. The reasons are fairly clear

- those involved have a clear incentive to use the new methods, which give them *more power with less pain*
- and at relatively modest cost that is more than made up for in increased effectiveness.

A Doing Mathematics	B Learning and Teaching	C System Change
Where to go? Why?		
lots of ideas + well tried experience	lots of ideas + a little experience growing in patches	very little sign; even calculators not integrated
How do we know?		
many examples work well	many effective examples	observation
How do we find out more?		
it will happen because participants have incentives	systematic small-scale studies in realistic circumstances + performance measures	enrichment packages + whole experimental curricula
How do we get it to happen?		
it will happen because participants have incentives	more support for systematic r & d	study dynamics of change + model experiments in realistic circumstances

Table 1

This progress is set out in some detail in the present volume, not only in the introductory chapters, but in the contributions of Steen and Stern, and to a considerable extent in the other articles. We shall therefore review it briefly and in general terms – the other chapters bring these generalities to life.

The changes in mathematics pervade the subject. The methods of first recourse in many areas are now numerical and, particularly, graphical, allowing an experimental approach with much less effort than in the past. Within mathematics most areas have been affected: discrete mathematics and combinatorics, number theory, algebraic and differential equations, finite groups, fractals and chaos, as well as all aspects of data analysis – these have all been profoundly changed; the papers in this volume describe some examples. The role of algorithmics is now central (see the chapter by Maurer). But equally clear is the effect on other aspects of pure mathematics – even the definitions of elegance and the status of proof.

Realistic applications to practical situations have suddenly become more accessible as the drudgery associated with realistic numbers and more realistic models is cut away. No longer is the focus on extracting the maximum from the few analytic models that are tractable by traditional methods.

Indeed, the role of such models in the future is an important issue for both mathematics and the curriculum. It seems likely they will continue to be centrally important, not as methods of solving problems, but as vivid illustrations of important effects. A closed algebraic expression displays, for those who can read it, the dependence on all the variables in the model – something which, if there are many variables, can be very difficult to communicate graphically or numerically. (For example, the expression for the response of a damped harmonic oscillator to a sinusoidal driving force depends on five variables – understanding the phenomenon of resonance from numerical solutions alone is not easy).

Several of the chapters that follow are focussed on the mathematical issues. Churchhouse's review ranges over the various fields and aspects of mathematical activity, looking at the effects of technology on the way mathematics is being done. Stern is mainly concerned with the impact of computer *science* rather than *technology* on the way mathematics is done while Steen introduces the perspective of the computer as a 'new mathematical species'. Maurer addresses what is, perhaps, the central area of change – the dominant place of algorithmic thinking and its implications.

Finally, a word of warning on *the student as mathematician*. Largely because of the *imitative* nature of the current curriculum, it is easy to get a quite false picture of students' capabilities. A mature mathematician has command of a range of concepts and techniques (or knows where and how to get such command) and uses them *autonomously* to express and manipulate ideas and relationships to get answers and understanding. There is clear evidence that, on such criteria, students' ***autonomous performance is several years at least behind their performance on imitative exercises.*** The calculator is a useful resource because teenage students can already use arithmetic for a range of purposes; in contrast it has been shown, for example, that even very bright 17 year-old students may not use algebra at all as an autonomous mode of expression, though they have had 5 years of success in manipulating it (Treilibs et al, 1981); so, for example, the benefits of a machine that will manipulate in a language they do not speak fluently are elusive, and maybe illusory.

The overall effect of these changes is well-summarised by Mascarello and Winkelmann, in a way that clarifies the challenge to designers of curriculum:

> "In total, there can be observed a specific shift in the spectrum of abilities, from precise algorithmic abilities to more complex interpretations, so to speak from calculation to meaning, which in a certain sense is a reversal of the historical evolution. In this process the mathematics to be mastered tends to become intellectually more challenging, but technically simpler."

Changes in mathematics education

As to the other *Domains, B and C*, nearly all the chapters that follow make suggestions for new curriculum elements based on these new methods of doing mathematics; readers will find many of these arguments stimulating, and even persuasive. Changes are surely needed and these suggestions seem better grounded than most.

Nonetheless, it must be recognised that such suggestions are ***fundamentally speculative*** at the level of large-scale implementation - by which we mean that converting them into a well-developed and tested curriculum for the typical teacher and the typical student is still a major challenge. This is the task of Domains B and C. We can have no reliable idea how far any suggestions we put forward will prove feasible in ***any***, let alone ***every***, educational system. Even if they are implemented reasonably faithfully, the full curriculum reality of what occurs will contain many surprising side effects; more likely, the translation from an idea to a small scale pilot experiment with exceptional teachers and facilities, and then to large scale reality will involve critical distortions of the aims of the exercise which may even, in the end, call into question its curriculum value. Thus rigour and vigilance are needed in this development process.

In case there are any who believe that we exaggerate the dangers, let me draw attention to a few famous examples of intended innovations in mathematical education which turned into something entirely different:

> The splendid Bourbaki enterprise was launched (believe it or not (Weil, 1979)) to establish a firmer foundation for mathematical education; few now see that as among the positive contributions it has made, while many are concerned at the effects of overemphasis on formalism that has arisen from this approach in school mathematical education.
>
> Smalltalk was originally devised by the Xerox Learning Research Group largely to produce a medium, the Dynabook, that would be 'as natural to a child as pencil and paper' (Goldberg, 1978); what has emerged is perhaps the most sophisticated graphics orientated data management system so far - an important achievement, but a very different thing. (The Learning Research Group was renamed the Software Concepts Group.) Smalltalk has not, at any rate, done any harm to the school curriculum, and among its offspring, the Macintosh microcomputer, may yet contribute notably in a quite different way.
>
> Our final example must be the reform movement in mathematical education of 30 years ago - 'new math', 'modern mathematics' and so on. Comparison of the initial aims agreed at conferences, the pilot schemes in a few exceptional schools, and the classroom reality of today shows the contrasts vividly. For example, in England the applications of mathematics occupied a central place in the original design; in most of the major courses that emerged applications were mentioned only to illustrate techniques with no serious attention to the practical situations involved. Equally, new mathematical concepts were introduced but often with none of the payoff that motivated their inclusion - because the serious examples originally envisaged proved too difficult for most students, and were replaced with trivial ones. The second wave of reform over the last decade has been rather more successful in remedying some of these defects, but has left others untouched.

These are cautionary examples to bear in mind when looking at new possibilities; it is not easy to realise the potential vitality of a field of mathematics in large-scale curriculum implementation. (One can easily imagine a vivid 'commercial' for the importance and excitement of a 'new' field like calculus; compare it with the reality of a typical introductory American college text.) That does not mean that vivid, effective implementation cannot be achieved – but success requires high-quality 'engineering' – and some poetry as well as the facts. (We have argued that, if the English language curriculum were like most mathematics curricula, the readings would be drawn entirely from the telephone directory.)

On the learning and teaching of mathematics

Progress in *Domain B* continues at a steady pace but, we would suggest, far too slowly to provide a sound comprehensive underpinning for the new curricula that we need *now*. Some of this progress is described in later chapters in this volume. These developments and the associated research represent deeper insights into the way technology can affect and enhance learning and teaching, together with some elements of curriculum that can be and have been used successfully in classrooms. They have rarely been tested on a large scale and thus represent only firm steps along the road towards the new curricula (Domain C). Let us begin by looking at some general effects in technology-related change.

In looking at curriculum reform, the first thing to note is the scale of it – perhaps 80% of current school classroom time is devoted to seeking fluency in a range of pencil-and-paper technical skills, all of which are now best done on computers of one kind or another. This we call *The Big Hole*.

Secondly, the swing towards teaching mathematics that is "intellectually more challenging, but technically simpler" takes both teachers and curriculum designers into areas outside the basis of their experience – thus such curriculum design should be essentially a research-based exercise, if it is to work well. It relates not only to content but to learning and teaching style. Everywhere the curriculum is still based on student imitation (e.g. HMI, 1977), dominated by:

teacher explanation +
illustrative examples +
imitative exercises.

This can lead to rapid apparent student progress, but much research evidence (see Bell et al, 1983) shows that the skills acquired are not reliably retained by most students, nor are they transferable – particularly to non-routine problems in the world outside the classroom. Fig. 1, for example, shows pre-, post- and delayed-test results of individual students from comparable groups taught by two different methods. The first 'positive only' method (a) is traditional explanation and reinforcement by practice; the second 'conflict' method (b) is based on students' discussion and 'debugging' of errors generated by them from their own misconceptions. The greatly improved long-term learning is stable across different topics (Bell and Basford 1989).

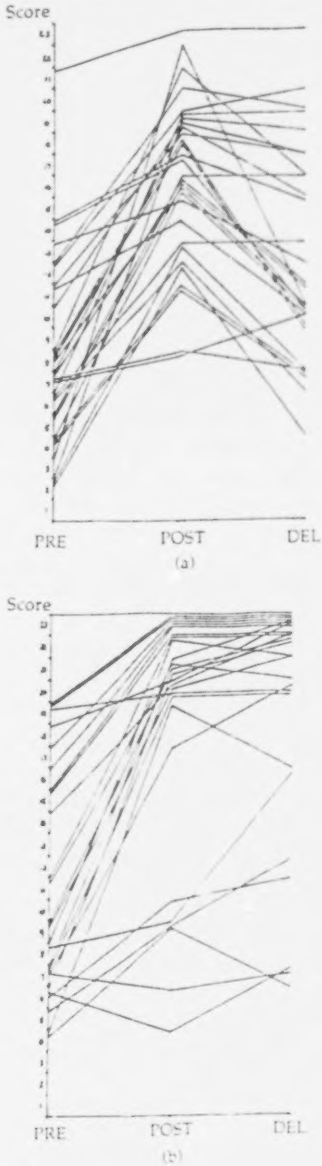


Figure 1

To achieve the flexible competence of understanding that the world requires, the pattern of classroom activities has to be widened to include some which give more autonomy, more initiative to the students. It is encouraging that the microcomputer has shown great promise in supporting such activities.

Thirdly, change is often threatening. Technology appears to reduce this threat, partly because it produces an obviously new situation and thus cannot imply criticism of the teachers' existing modes of operation. This more than compensates for the extra barrier of learning to use the equipment – provided it is reliable. Further, recent work on the teaching of non-routine problem solving of a wide variety of kinds shows the importance of the strategic skills of comprehension, modelling, interpretation and evaluation – just the skills that are brought to the front of our attention by the computer.

In summary, the two great springs for change in mathematical education in the past and the next decade are technology and autonomy. Fortunately, they can help each other, though there is much to be learnt as to how best it might be done.

However, conversely, we believe that it may be important not to discount too easily the value of traditional skills, remembering that the current generation of innovators have the 'traditional' background, as well as newly acquired skills with computers. Almost certainly, much of what was learnt is useless but we need to check for losses as well as gains in a curriculum change. Mental facility with numbers, graphs and expressions has always been an asset. What is its status now?

Exploratory investigation as a key element in the curriculum has been a major objective in English mathematical education for at least 30 years – the Association of Teachers of Mathematics was founded largely to promote it; in the USA, we know that it has been a focus since Polya (1945) and officially central at least for a decade, since the NCTM Yearbook on problem solving. However, despite strenuous efforts it has not become a regular part of the curriculum anywhere except in a tiny minority (less that 1 percent) of classrooms. We have a lot of evidence and some understanding of how difficult such activities are for the typical teacher to handle in the classroom; appropriate support must be developed. Everyone rightly emphasises the curriculum opportunities for exploration, for 'experimental mathematics', that the computer provides; however, the development of such an investigative element in the curriculum will succeed only if it confronts the difficulties such activities present for teachers.

Equally, the challenge to explore must be at a level matched to the student – if the aim is to 'discover' in an hour or so some important mathematical achievement that took a genius half-a-lifetime to create, the exploration will have to be so closely guided as to be essentially fake; on the other hand, interesting, though less global, problems which students can tackle autonomously on their own resources, do exist at every level. For example, programming projects, at school and university, have shown some of the possibilities, and the difficulties for the teacher. A creative and systematic program of detailed empirical development will be essential if exploration is not to degenerate in most classrooms into that closely guided 'discovery learning', which is really an alternative style of explanation. The computer can, of course, help.

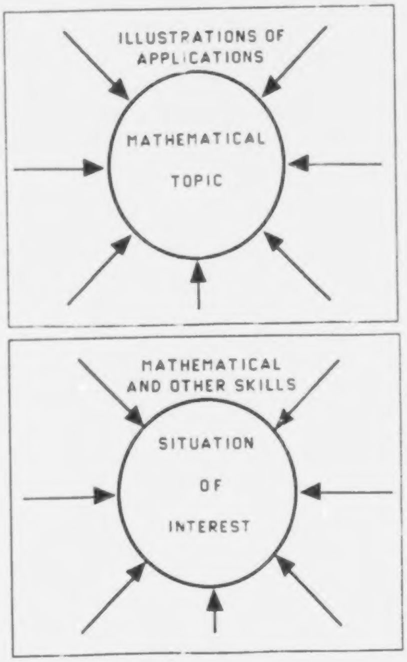


Figure 2

The emphasis on problem solving encourages applications of mathematics – even some with real data. It is important to note two different kinds of application, the illustrative and the situational (see Fig. 2, from Swan 1990). In illustrative applications the focus is really on the particular mathematical topic; the applications are there in support – to help conceptual understanding through concrete illustration, to show how mathematics can be applied, and to provide practice. In realistic, practical situations from outside mathematics the position is quite different – in principle, any or all of the mathematics you know could help you to tackle the problem, along with other knowledge and strategic skills. Both these kinds of application are needed – but the student must know which 'game' is being played, because the best tactics are quite different. For illustrative applications the aim is to show how much mathematics you know; for situations, it is to provide the most powerful understanding of the practical problem.

Finally, assessment – in arriving at a curriculum, assessment can be very helpful in clarifying curriculum definitions, particularly by example. ***What range of types of tasks do we want our students to be able to do?*** It can be argued persuasively that a task-defined curriculum has great advantages over a 'scope and sequence' approach, though the two are complementary – one being synthetic and the other analytic.

The chapters that follow range widely over this field, complementing analysis with the essential vivid exemplification which we have had to exclude from this overview. Ralston's review takes a look without prior assumptions ('zero-based') at what the curriculum should contain. The criteria include value to the student and, related, the way mathematics is done nowadays together with lessons from the psychology of learning. Cornu's review covers a wide range of roles for the computer in enhancing teaching and learning – Computer-assisted Learning is as diverse as Paper-assisted Learning. The remaining chapters that follow present a kaleidoscope of key aspects of computer use in learning and teaching. Tall and West explore and illustrate the visual aspects of learning and the contributions that computer graphics can make. Hodgson and Muller look at the other major shift enabled by the technology – automated symbolic manipulation. Graf, Fraser, Klingen, Stewart and Winkelmann take a broader look at the various modes of use of technology in school and the potential of each in the elementary school; both computers and calculators are discussed. Steen takes a similar approach to college mathematics – where the issues range from the place of discrete mathematics to computer literacy for all students. Seidman and Rice, and Mascarello and Winkelmann look in a most stimulating and practical way at a related central problem – the integration of discrete and continuous mathematics within a college course.

All of these show what has been achieved in Domain B – in realising, on a pilot scale, something of the enormous potential of computers and informatics in enhancing the learning and teaching of mathematics. They also set targets for future development.

New mathematics curricula

The lack of progress in *Domain C* is the major mismatch between intentions and outcomes over the last seven years. It is notable that even the use of simple calculators has not been fully integrated into the curriculum in any country in a way that realises their known potential for enhancing mathematical performance (even on traditional skills!).

The reasons are less clear than is sometimes thought by those who ascribe it simply to teacher inertia and/or parental opposition. Certainly, teachers and other educators do not have the direct incentives that the use of the technology provides for those doing mathematics. It does not so obviously promise to increase their power as professionals, or to make their lives easier or more rewarding. Rather, it makes obsolete a large part of the standard professional work of mathematics teachers and threatens them with, at least, a need for new skills, both mathematical and pedagogical. Equally parents and others tend to believe that their own education remains valid – after all, look what it has done for them!

However, when one compares the support offered to teachers to make these changes with that which they routinely receive simply to sustain the current curriculum, the contrast is stark. The textbooks and other materials are not comparable, or often even available. Retraining is sparse, as is coherent explanation of what is being attempted, and why. The temptation to blame the lack of change on teachers is not only misguided but fruitless – they are who they are. It is up to those who seek change to find, and to deliver, an effective and appropriate mixture of pressure and support.

We have evidence that teachers actually welcome change, provided they are confident that the pace and level of support is such that they can cope with it without undue effort. As with many profound curriculum changes, systems have so far failed to provide any basis for such confidence.

In their chapter Cornu and Balacheff look at the problems of the new pedagogy and how it may be communicated to teachers in training, a key area in Domain C. We already have evidence (see Burkhardt, 1984, 1985) that the potential of the microcomputer for helping teachers to enhance student learning presents a tremendous opportunity for curriculum enhancement. The effects on the dynamics of the classroom can be profound, but they are often subtle; for this reason there is a great deal still to do before we have even a broad understanding of what can happen in the various modes of computer use in education.

We shall illustrate the sort of thing that may be expected by describing one application that has been developed and studied in some detail, and which has proved particularly rich – the use by the teacher of a single micro in the classroom, programmed to be a 'teaching assistant'. We do so for various reasons: It is less familiar to most people; it brings out some general points about the overwhelming importance of the people, teacher and pupils, and of the dynamics of their interaction; and it is particularly relevant to schools as we know them because it seeks to enhance the performance of a teacher working with a group of children in the classroom in the normal way. It also only requires one microcomputer per class rather than one per child.

This mode of use, set out by one of us (Fraser, 1981), has been shown to have remarkable effects in leading typical teachers in a quite unforced and natural way to broaden their teaching style to include the 'open' elements that are essential for teaching problem solving (Fraser et al, 1983, 1988). Since this is a crucial aim that reformers have been trying to achieve for at least thirty years with little or no effect, this is a valuable result. It is worth explaining briefly why these effects come about. First, the micro is viewed by the students as an independent 'personality'. It takes over for a time a substantial part of the teacher's normal 'load' of explaining, managing, and task setting. These are key roles played by every mathematics teacher. The micro takes them over in such a way that the teacher is led into less directive roles, including crucial discussion with the children on how they are tackling the problem, providing guidance only of a general strategic kind – counselling if you like.

These principles have been incorporated into a range of teaching materials that enable typical teachers to sustain in their classrooms, without exceptional effort, these learning activities of a more open kind. This book is full of other examples of curriculum components that have been shown to work well in typical circumstances, or can be developed to do so. It is also important to recognise that there will be disappointments – or at least frustrations – in the development process. In the last few years there has been further progress to report in the thorough, and imaginative, development of substantial curriculum elements. They illustrate what can be done. The *Journeys in Mathematics* project (EDC, 1991), funded by the National Science Foundation, has developed a series of modular units that exploit the potential of computer support for the elementary school mathematics classroom in a variety of powerful ways. Similarly, *The Power Series* (UCSMP/Shell Centre, 1992) offers an effective element in teacher development, through the support that the 'single micro classroom' can provide in exploring new, more open ways of working. There are now many other 'enrichment materials' using the computer to support learning, particularly those less-routine activities that many teachers find difficult to handle.

Full technology-integrated curricula, with materials to support them, are hardly available yet. If there is an exception, it is a few new courses in higher education, such as that described by Hodgson and Muller. There are early signs of moves to develop materials to support complete curricula; some of the latest round of NSF-supported projects, for example *Seeing and Thinking Mathematically at EDC* and a parallel project at TERC, have a strong emphasis on technology.

It is interesting that all three examples quoted above (and many others) use the computer as a 'catalyst for learning' (Fraser, 1989) rather than as a 'tool' for doing mathematics or a 'tutorial system'.

No one doubts that the computer as a tool is a central element in the curriculum but the development of curricula to realise this is slow. Of course, the level of computer provision needed to make it more than a passing experience is still beyond most schools. It is interesting and ironic to remember the pioneering work of the Computer Assisted Mathematics Program (Johnson et al, 1966-68), in which students learned mathematics in a Basic programming environment; Kieren (1974) showed that far more students got through to fluency in algebra in this way – a result that has been confirmed but not yet implemented anywhere. As we have noted, even the simple calculator is far from fully integrated into curricula.

Computer-based tutorial systems have continued to emerge, and to become more sophisticated, sometimes embodying elements of artificial intelligence of an 'expert system' kind. Apart from programming itself, perhaps the first big idea for using computers in mathematical education was in teaching technical skills, particularly arithmetic. The approach followed the behaviourist teaching-machine model. To provide effective teaching in this way has proved a much harder problem than was expected. We believe that it is still far from solution. It seems that the computer-tutor can be effective in teaching facts and straightforward techniques to people who have little difficulty with them; so, of course, are other methods. However, despite great efforts by some talented people, it has not so far proved possible to write programs which are successful in diagnosing and remediating students' conceptual errors underlying technical skills that they find difficult. It remains true that tutorial systems have not begun to tackle the main defects of the traditional (and largely current) mathematics curriculum, still concentrating on automating those learning activities (largely drill-based) that are both over-represented and, on their own, ineffective.

The next steps

It seems from the above that Domains A and B are making steady progress and that Domain C presents the greatest difficulties – thus it seems that further work on large-scale implementation should become a priority over the next decade. There are some small signs of movement in this direction in various countries. However, the difficulty of achieving large scale change of any kind is often underrated, or at least neglected. It clearly needs empirical study of the dynamics of change in the education system as a whole, with all the factors this brings in. We already know far more about the benefits that could flow from the use of technology (even within current financial constraints) than is realised in practice. Without attention to Domain C, this mismatch will simply get worse.

What, specifically, are we to do about this? This is not the place for a serious discussion of methodologies of research and curriculum development (Burkhardt, Fraser and Ridgway, 1990). Very briefly, there is no proven successful answer but some seem to be less susceptible to corruption of outcomes than others. We believe that the essence is an empirical approach – find out what actually happens to your draft ideas in practice, in circumstances sufficiently representative of what you are aiming for, and then revise the materials repeatedly until they work in the way intended. We have found (e.g. Shell Centre, 1984) that such an approach, taken for granted in other fields, can combine educational ambition with user-friendliness to a level not achievable with more casual development. Structured classroom observation makes a key contribution to this approach, providing much richer feedback than is often acquired in the development of educational materials (Burkhardt, Fraser et al, 1982). However, more rigorous comparative evaluation of alternative approaches is sorely needed. A few more comments are made below.

It is important to ask of everyone in the system, but particularly teachers, "Why should they change?". It seems (Fullan, 1980) that both pressure and support are needed for effective change but producing a balanced 'well-engineered' package that works is still an unsolved problem. One lever for change that cannot be ignored is the assessment system; if a system does not recognise, measure and reward new curriculum elements then they will not be taken seriously by many – WYTIWYG (what you test is what you get). Pressure is often preferred by politicians because it is less expensive than support, so that an effective balance is destroyed.

The questions we have raised imply a great deal of work, integrating research techniques with curriculum development, before we have even a basic understanding of the classroom potential that we see so vividly illustrated in this book. Experience suggests that, along the way, we shall find other possibilities of at least as much promise.

In order to realise any of these possibilities, they will need to be systematically developed in detail with representative samples of teachers and students, using structured detailed data from the classroom.

Systematic research and development

The slow progress in B and, particularly, C partly reflect a general problem in education – that the level of expenditure on designing and developing soundly-based changes is remarkably low; in England and the US, for example, this 'research and development ratio' is substantially less than 0.1% of educational expenditure, whereas in other changing fields such as medicine or modern industry it is typically *between 5 and 15%*. We believe that this arises because education is still dominated by the 'craft-based' approach, which assumes that experienced professionals have satisfactory methods of handling each situation that presents itself – that everything is basically well under control.

This craft-based approach works well when two conditions are satisfied:

the system is working satisfactorily
and
there is no expectation of major change

Otherwise it involves the extrapolation of reliable experience beyond its domain of validity – always a hazardous process.

In this respect the situation in education is rather similar to that in medicine a century ago, or engineering further in the past. There are signs that the more systematic 'research-based' approach (which now dominates the other two fields) is more widely recognised as important in education, but there is still a long way to go. No one would dream of using a drug, or flying in an aeroplane, that had only been developed and tested as sketchily as most new (or, indeed, old) curricula. Nonetheless, the situation persists – perhaps because educational disasters are much less immediately visible.

Some dismiss such arguments on the grounds that teaching and learning are much more varied and less controllable than engineering or medical situations. Though this is true in some respects, systematic observation of typical mathematics classrooms (HMI, 1979) shows a remarkable uniformity in delivering a curriculum that is inappropriate and seriously impoverished by current standards. Similar results are observed in most countries.

The research and development ratio illustrates how little serious attempt has been made in education to work systematically to do better. The processes of systematic development are either absent or sketchy, particularly in the quality of feedback and the typicality of the development environment. The search for better development methods is almost non-existent. We believe that we are still essentially leaping off cliffs, flapping 'wings' tied to our shoulders – but nobody notices the mess.

These things need not be. So far from costing money, they represent a potential source of large saving in the overall operation of the education system.

These considerations are not, of course, confined to technology-inspired change. They are, however, particularly acute in that circumstance because of the pace of change needed. We are facing a situation in which children are taught to do mathematics in ways that are very largely outmoded, with at least 80% of curriculum time wasted on trying, more or less successfully, to develop fluency in skills of now-limited value.

In planning a systematic approach, we think it is useful to distinguish different levels of research and development in education. Studies at each successive level involve an order of magnitude more students, and teachers, than at the previous one. Four levels are:

L Learning – studies of student's learning, the nature of cognitive processes, difficulties and misconceptions (10^0 - 10^1 children minimum)

T1 Teaching Possibilities – studies of different kinds of stimuli and their effects on student learning (10^1 - 10^2 children minimum)

T2 Realizable teaching – studies on what can actually be achieved with typical teachers under realistic circumstances (10^2 - 10^3 children minimum)

C Curriculum change on a large scale – studies of how curriculum change can be effected and what other school or social factors affect it? (10^4 - 10^7 children minimum)

All these levels are important, the earlier ones contribute to a fundamental understanding of the later ones. However, much more serious work has been done at the early L and T1 levels; a more balanced effort would be productive. The crucial distinction between T1 and T2 is often not made. At the T1 level the teacher variables are almost irrelevant – work there simply shows that there is a teacher, usually a member of the development team, who can make these things happen. At the T2 level, curriculum developers face the challenge of showing that a wide range of unexceptional teachers, in normal circumstances of support, can also function in the desired ways. Similarly, the C level brings in all the variables that relate to the pressures on the classroom from school and society, which are so critical to the implementation of any change. These distinctions appear to be important in the limited impact of technology so far – and in doing better in the future.

Acknowledgements

We have benefited from many conversations with out friends in the ITMA Collaboration, particularly Richard Phillips, Jan Stewart, Jim Ridgway and Jon Coupland, and others around the world, particularly David Tall and Tony Ralston.

REFERENCES

Bell A.W., Costello J. and Kuchemann D. [1983]: *A Review of Research on Mathematical Education, Part A: Research on Learning and Teaching*, Windsor: NFER- Nelson.

Bell A.W. and Basford D. [1989]: *A conflict and investigation teaching method and an individualised learning scheme – a comparative experiment on the teaching of fractions*, Nottingham: Shell Centre for Mathematical Education; also *Teaching for the Test*, Times Educational Supplement, 27 Oct.

Burkhardt H., Fraser R. et al [1982]: *Design and Development of Programs as Teaching Material*, London: Council for Educational Technology.

Burkhardt H. [1984]: *How can micros help in schools?: research evidence*, Nottingham: Shell Centre for Mathematical Education.

Burkhardt H. [1985]: The Microcomputer: Miracle or Menace in Mathematical Education in *Proceedings of ICME5* (M. Carss, Ed.) Berlin: Birkhauser.

Burkhardt H., Fraser R. and Ridgway J. [1990]: The Dynamics of Curriculum Change in *Developments in School Mathematics Around the World, Vol 2* (I. Wirszup and R Streit, Eds.), Reston, VA: National Council of Teachers of Mathematics.

EDC [1991]: *Journeys in Mathematics*, Scotts Valley, CA: Wings for Learning.

Fraser R. [1981]: How to use Homo Sapiens in a Computer Environment in *Proceedings of the 1980 ADV Conference*, Vienna: ADV; also *Design and evaluation of educational software for group presentation in Microcomputers in Secondary Education* (J. Howe and P. Ross, Eds.), London: Kogan Page.

Fraser R., Burkhardt H., Coupland J., Phillips R., Pimm D. and Ridgway J. [1983, 1988]: *Learning Activities and Classroom Roles*, Nottingham: Shell Centre for Mathematical Education and *J. Math. Behaviour*, 6, 305-338.

Fraser R. [1989]: *Introduction to Computers and the Teaching of Mathematics* (E. Dubinsky and R. Fraser, Eds.), Nottingham: Shell Centre for Mathematical Education.

Fullan M. [1980]: *The Meaning of Educational Change*, New York: Teachers College Press.

Goldberg A. [1978]: *Trends in Hardware and Software* in *Informatics and Mathematics in Secondary Schools: Impacts and Relationships*, 1977 IFIP Conference (D.C. Johnson and J.D. Tinsley, Eds.), Amsterdam: North Holland.

HMI [1979]: *Aspects of Secondary Education in England, Report of the HMI Secondary Survey*, London: HMSO.

ICMI [1984]: The Influence of Computers and Informatics on Mathematics and its Teaching, *L'Enseignement Mathmatique*, 30, 159-172.

Johnson D.C. with Hatfield, L., Walther, J., La Frenz, D., Katzman, P. and Kieren, T. [1966-68]: *Computer Assisted Mathematics Program*, Glenfield, IL: Scott Foresman.

Kieren T [1978]: *Informatics and the Secondary School Mathematics Curriculum* in *Informatics and Mathematics in Secondary Schools: Impacts and Relationships*, 1977 IFIP Conference (D.C. Johnson and J.D. Tinsley, Eds.), Amsterdam: North Holland.

Shell Centre [1984]: *Problems with Patterns and Numbers, a Module of the Testing Strategic Skills Programme*, Nottingham: Shell Centre for Mathematical Education.

Swan M. [1990]: Mathematical Modelling for All Abilities in *Proceedings of ICTMA4*, Chichester, U.K.: Ellis Horwood.

Treilibs V., Burkhardt H. and Low B. [1981]: *Formulation Processes in Mathematical Modelling*, Nottingham: Shell Centre for Mathematical Education.

UCSMP/Shell Centre [1992]: *The Power Series* (six modules for the single micro classroom in the elementary school), Chicago: University of Chicago School Mathematics Project and Nottingham: Shell Centre for Mathematical Education.

Weil, A. [1979]: *History of Mathematics* in *Proceedings of the 1978 International Congress of Mathematicians*, Helsinki: ICM.

An Update of the 1985 Strasbourg Conference

The first edition of this book grew out of a conference in Strasbourg in March 1985. The attendees at that conference divided themselves into three Working Groups on the subjects of **The Effect of Computers on Mathematics, The Impact of Computers and Computer Science on the Mathematics Curriculum** and **Computers As an Aid to Teaching and Learning Mathematics**. The reports of these three working groups formed the first three chapters in the previous edition. In this edition the leaders of the three workshops have updated the reports which appeared in the previous edition. These updated reports appear on the following pages.

Part I

THE EFFECT OF COMPUTERS ON MATHEMATICS

R. F. Churchhouse
University of Wales, Cardiff, UK

1.0 Introduction

Mathematical concepts have always depended on methods of calculation and methods of writing. Decimal numeration, the writing of symbols, the construction of tables of numerical values all preceded modern ideas of real number and of function. Mathematicians calculated integrals, and made use of the integration sign, long before the emergence of Riemann's or Lebesgue's concepts of the integral. In a similar manner, one can expect the new methods of calculation and of writing which computers and informatics offer to permit the emergence of new mathematical concepts. But, already today, they are pointing to the value of ideas and methods, old or new, which do not command a place in contemporary "traditional" mathematics. And they permit and invite us to take a new look at the most traditional ideas.

Let us consider different ideas of a real number. There is a point on the line R, and this representation can be effective for prompting the understanding of addition and multiplication. There is also an accumulation point of fractions, for example, continued fractions giving the best approximation of a real by rationals. There is also a non- terminating decimal expansion. There is also a number written in floating-point notation. Experience with even a simple pocket calculator can help validate the last three aspects. The algorithm of continued fractions - which is only that of Euclid - is again becoming a standard tool in many parts of mathematics. Complicated operations (exponentiation, summation of series, iterations) will, with the computer's aid, become easy. Yet even these simplified operations will give rise to new mathematical problems: for example, summing terms in two different orders (starting with the largest or starting from the smallest) will not always produce the same numerical result (*see*, e.g., Churchhouse, 1980, 1985).

Again, consider the notion of function. Teaching distinguishes between, on the one hand, elementary and special functions - that is, those functions tabulated from the 17th to the 19th century - and, on the other, the general concept of function introduced by Dirichlet in 1830. Even today, to "solve" a differential equation is taken to mean reducing the solution to integrals, and if possible to elementary functions. However, what is involved in functional equations is the effective calculation and the qualitative study of solutions. The functions in which one is interested therefore are calculable functions and no longer only those which are tabulated. The theories of approximation and of the superposition of functions - developed well before computers - are now validated. The field of elementary functions is extended, through the discretisation of nonlinear problems. Informatics, too, compels us to take a new look at the notion of a variable, and at the link between symbol and value. This link is strongly exploited in mathematics (for example, in the symbolism of the calculus). In informatics, the necessity of working out, of realizing the values has presented this problem in a new way. The symbolism of functions is not entirely transferable, and the attributes of a variable are different in languages such as Fortran, Lisp and Prolog.

In the sections that follow we look at some aspects of how computers and informatics have already affected mathematics and mathematical research and present some thoughts on what future effects might be seen. We do not claim that our survey is comprehensive, especially so in the disciplines of applicable mathematics, but we hope that it provides some pointers. In any event information technology, in the widest sense, is advancing far too fast for any predictions to be of value for a period of more than a few years.

1.1 New and revived areas of mathematical research

Computers not only provide a new tool in mathematical research and teaching. They are, at the same time, themselves the source of new areas of research. Not all of the research stimulated by the availability of computers is in new branches of mathematics; some is of ancient lineage, going back to the 19th or 18th century, but open now to attack with a weapon not available to Euler, Gauss, Jacobi, Ramanujan, etc. Who can doubt, though, that these giants of the past would have exploited these new possibilities with enthusiasm had they been available? It is one of the unique features of mathematics that it is based upon a body of results that never loses its value. Fashions and interests may change, but the neglected subject of the last century, or even of the last millennium, may prove to be of new interest at any time when conditions are right for its re-emergence. So the corpus expands; nothing ever dies, though it may remain dormant for centuries. In the age of information technology we need to emphasize this fact, for it underlies everything that follows.

One of the most famous examples of mathematical research being stimulated by the use of a computer is the soliton (solitary wave) solution of the Korteweg-de Vries equation by Zabusky and Kruskal (1965), which was initially suggested by numerical results. Continuing experimental investigations have indicated the existence of other, related, solutions and theoretical research has provided a substantial framework for investigating soliton solutions of several nonlinear wave equations.

Another example is found in the work of Yamaguti, which may be summarized briefly by saying that he observed continuous, but nowhere-differentiable, functions via numerical experiments on dynamical systems defined iteratively whose solutions exhibit very chaotic behaviour. Particular cases produce the Weierstrass function and the Takagi function (see also the chapter by Tall and West); the latter may be written

$$T(X) = \sum_{k=1}^{\infty} 2^{-k}\phi^{(k)}(X)$$

where

$$\phi(x) = \begin{cases} 2x & 0 \le x < 1/2 \\ 2(1-x) & 1/2 \le x \le 1 \end{cases}.$$

and has recently been used in teaching elementary analysis. Further research, in collaboration with Hata on a family of finite difference schemes led to Lebesgue's Singular Function.

Among long-established branches of Pure Mathematics where computers have had a major impact are Group Theory, Combinatorics and Number Theory. Many applications of computers in these areas have been published in proceedings of conferences (for example, Churchhouse and Herz (1968), Atkin and Birch (1971), Leech (1970)).

The applications are already too numerous to list in full or describe in detail but it is clear that the search for sporadic groups, the investigation of Burnside's problem, the study of rational points on elliptic curves, and the search for large primes would be quite impossible without computers. The factorisation of large integers is another example; although intrinsically it is not an exciting topic it has recently assumed considerable importance in relation to cryptography and public-key systems (Beker and Piper, 1982). Many of these applications have benefited considerably from the availability of program packages specifically designed as an aid for researchers in the field; the CAYLEY system for the study of finite simple groups is a well-known example. Another is the development of Symbolic Mathematical Systems (*see* Section 1.6 and the chapter by Hodgson and Muller). Such systems relieve research workers of a great deal of drudgery. Indeed, they make possible manipulations which just could not be done manually in any reasonable time or with any valid hope of an accurate result. Another "old" topic that has taken on a new lease of life is that of continued fractions, both as providing approximations to real numbers and, in analytical form, in numerical analysis.

The availability of colour graphics displays and packages has opened up exciting possibilities for research not only in geometry, modelling and fluid flow but in less obvious areas such as analysis (*see* the chapter by Tall and West). The study of the iteration of complex-valued functions has been transformed recently; the complex nature of Julia sets and their descendants is made beautifully apparent by the use of colour graphics, even through much of their mathematical nature remains unknown (see, for example, Section 1.4 below).

It is clear to us that the computer is having, and will continue to have, a significant impact on the directions of mathematics research and on the way in which mathematicians carry out their research. Computers will not only be commonly used to arrive at conjectures but also to assist in finding proofs. In addition, some important questions are raised: (i) How should computers be used to assist mathematicians in communicating their discoveries and in keeping abreast of the research of others? and (ii) What are likely to be the intellectual and social consequences, so far as mathematics and mathematicians are concerned, of the widespread interest in, and use of computers?

1.2 Proof

In mathematics a "proof" is, strictly, a chain of deductions from the axioms; in practice, of course, a proof is accepted if it makes use of results which have themselves been deduced from the axioms, or from other results, etc., etc. It would be possible, but exceedingly tedious, to write out a proof of the theorem that every positive integer is the sum of the squares of four integers by starting from the axioms of arithmetic, but few people would regard this as necessary and would accept various intermediate steps - an identity of Jacobi, or representation of integers by binary quadratic forms - as valid rungs on the ladder, since each of these steps is deducible from other results which are deducible ..... from the axioms.

Computers might be used in mathematical proofs; they might, initially, suggest what is true and, equally important, what is not, they might be used for computations which are required in a proof; they might be used - as in the proof of the 4-colour theorem (Appel and Haken, 1976) - to examine all of the finite set of cases, on which the truth of the theorem ultimately depends; they might even be programmed to find part of the proof by trying many possible combinations of known axioms, theorems or identities, though "combinatorial explosion" makes such an approach infeasible except in very special cases.

As examples, computers have been used to suggest results in group theory, combinatorics, number theory, coding theory and to support the truth of conjectures such as the Riemann Hypothesis. For an early survey article *see* Churchhouse (1973). Among notable theorems which were initially conjectured on the basis of numerical evidence are the Prime Number Theorem (Gauss) and several important results of Ramanujan (1927) including the congruence properties of the partition function and of the function $\tau(n)$. On the other hand Lander and Parkin (1967) and a computer found that

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$

and so *disproved* a conjecture of Euler that had stood for nearly 200 years. One very specific recent achievement deserves special mention viz: the disproof by Elkies (1988) of the Euler Conjecture on sums of fourth powers. Euler conjectured that (inter alia) no fourth power could be the sum of three fourth powers. Elkies however found that

$$(2682440)^4 + (15365639)^4 + (18796760)^4 = (20615673)^4$$

and went on to prove that not only are there an infinity of such counterexamples but, when expressed as the representation of 1 as the sum of three rational fourth powers, the solutions are dense in $< 0, 1 >$.

Accuracy and reliability of the computations should not be an issue today. Where a result is sufficiently important or in doubt it can be checked by someone else on a different machine; this has been done on several occasions and if the result is confirmed and, assuming that the underlying mathematics is correct, the result can be accepted with considerable confidence, if not certainty. Computer-assisted proofs need not be any more suspect than purely human proofs; many false "proofs" - including some of the 4-colour theorem - have been published in the past; we do not believe that the computer will increase the number of false proofs, quite the contrary.

It is, of course, accepted that no amount of numerical evidence constitutes a proof of a theorem relating to an infinite set; the numerical evidence may be misleading even for a very large set of values of the variables involved. A well-known example from analytic number theory is Littlewood's proof (*see* Ingham, 1932) that despite all the numerical evidence then, and even now, available

$$\pi(x) - \int_2^x \frac{dt}{\ln t}$$

(where $\pi(x)$ indicates the number of primes less than or equal to x) not only eventually changes sign, but does so infinitely often.

A criticism of computer-assisted proofs - such as that of the 4-colour theorem - is that they tend to rely on brute-force and give little insight into why the theorem is true. Unfortunately some results e.g. finding large primes or factoring large integers intrinsically require such methods, and whilst it may be true that a computer proof may bring little insight, its very existence may inspire people to find more elegant, shorter, or illuminating proofs.

Taking a longer-term view, the availability of computer assistance may encourage mathematicians to a more precise syntax and to express more formally what is in their minds (de Bruijn, 1970). Such a development may, in turn, aid the teaching of the art of constructing proofs and so lead to the development of "expert systems" to undertake a least some aspects of mathematical work (including all the routine algebraic manipulation, computation, etc.), in partial fulfillment of Leibniz's dream of a rational calculating device.

One final point: Since every proposition that is provable has among its many proofs one of minimal length and since the proofs of any given length are (at most) finite in number there must be true theorems of mathematics that cannot be demonstrated by traditional discourse within the longest human lifetime. It would appear then that there are mathematical theorems that can *only* be proved with the aid of computers if we are unwilling to wait too long.

1.3 Experimentation in Mathematics

Certain branches of mathematics have always been open to experimentation but the arrival of computers means the scope for experimentation in mathematics has been greatly increased. In some of the sections above we have indicated cases where experiments have been used to provide data on which conjectures and, in some cases, theorems have been based. Euler, remarking on the necessity of observation in mathematics, said: "The problems of numbers that we know have usually been discovered by observation, and discovered well before their validity has been confirmed by demonstration ....."

The sheer speed of computers means that calculations which would once have taken a lifetime can now be completed in hours, or even minutes. Add to this the fact that the results can often, if required, be presented in graphical form rather than as a list of numbers and we see that the interpretation of the experiments may be made much easier. The case of the iteration of complex-valued functions illustrates this point.

Of course, when a constraint is relaxed, there is a danger of excess. The ability to perform calculations does not mean that everything can or should be calculated. There is a balance to be struck and this must be guided by experience - not to mention the cost of the computations. The effort and cost involved need to be combined with the probability of success, in the sense of solving a problem or uncovering some useful fact. Computation for the sake of computation is not to be encouraged.

Although experimentation in pure mathematics has its uses it is, perhaps, in the area of statistics that it is particularly valuable. We take two examples.

Simulation

Even before the availability of the modern computing technology, experimental sampling and Monte Carlo methods have played a role in statistics for studying the performance of statistical techniques under the assumption of probability models. The computer has enhanced this aspect on a large scale. One famous example is the Princeton Robustness Study (Andrews et al, 1972) where sets of estimators under a system of different modelling assumptions are studied by means of computer simulation. The results have stimulated new mathematical research into robust estimators (e.g. asymptotic theory) but on the other hand they cannot merely be interpreted as conjectures that can and should be validated by mathematical proof, but they have an importance in itself and have already influenced the practice of analyzing data.

More generally, computers have given a major impetus to the idea of *mathematical modelling* wherein a physical or logical situation is embodied in a mathematical model whose operation may then be simulated on a computer. Thus we no longer need to place physical models in a wind tunnel but instead simulate the model on a computer. Similarly we do not have to build a new telephone system to see if it works since we can first simulate the system on a computer.

Exploratory Data Analysis

It is sometimes stated that the computer has led to an unwelcome shift from hard thinking to a senseless computation of examples and experimentation. A balanced picture would say that the computer has led to broader variety of "types of rationality" to approach problems and it is necessary to judge in every situation which approach is more reasonable.

The classical paradigm for applying statistics is to think first very hard and then construct a probabilistic model and an adequate design for gathering data. But this strategy is not feasible in quite a lot of situations where little is known about the data and the underlying system of interest. In connection with the numerical and graphical capabilities of computers a new methodology of data analysis, called Exploratory Data Analysis (Tukey, 1977), has been developed. The computer has made it possible to experiment with several models for a data set, to construct a variety of interesting plots of the data to gain insights into patterns, structures and anomalies of the data and to develop conjectures concerning the features of the system underlying the data. Such a type of exploratory mathematics would not be practicable on a large scale without using computers.

1.4 Iterative methods

Methods of solving systems of linear equations are traditionally divided into (i) direct and (ii) indirect, or iterative, methods. The direct methods include Gaussian elimination, the indirect methods include the Gauss-Seidel method. Direct methods have the advantages (a) that they will always produce the solution provided that it exists, is unique and that sufficient accuracy is retained at every stage, and (b) that the solution is found after a known number of operations. They have the disadvantage that very sparse systems of equations, such as arise in finite difference approximations to differential equations, may become rapidly less sparse as the elimination process proceeds so raising the storage requirement from a multiple of n (for n equations) to something like n^2. Iterative methods, on the other hand, may fail to converge to a solution and, if they do converge, it is not obvious how many operations they will require to produce the desired accuracy. They have, however, the very considerable advantages that they are very well suited to computers and preserve the sparsity of the coefficient matrix throughout.

Direct methods of solution of *nonlinear* systems are rarely available; there is, after all, no direct method for solving the general polynomial of even the fifth degree and so iterative methods are generally used. As in the case of linear systems, convergence may not always occur, though conditions sufficient to ensure convergence are usually known; and although in some cases the number of iterations necessary to produce convergence to a specified accuracy may not be easily predicted, it is frequently not a matter of great importance and, if time is limited, accelerating techniques can often be used.

The revival of interest in iterative methods brought about by the use of computers has led to significant advances in the study of functions which are iteratively defined, e.g. by a relation of the type

$$Z_{n+1} = F(Z_n)$$

where Z_0 is a given complex number and the function $F(Z)$ may contain one or more parameters. Some functions of this type, such as

$$Z_{n+1} = Z_n^2 + C$$

were studied over 60 years ago by Julia (1918) and Fatou (1919), but attracted relatively little interest at that time. In the case where the function $F(Z)$ involves one complex parameter C and we define the set of points K_C to be those points Z such that the iterated sequence of points given by

$$Z, F(Z), F(F(Z)), \ldots \text{etc.}$$

does *not* go to ∞, then the boundary of K_C is called the *Julia set* associated with $F(Z)$ and C. Only recently, thanks to the availability of computers and, particularly, of colour graphics terminals has the extraordinary nature of these Julia sets and their numerous spin-offs been appreciated. For example, the Mandelbrot set is defined as the set of values of C for which K_C is connected. The relation above is a *fractal* curve, the discovery of which, due to Mandelbrot, has inspired a great deal of exciting and attractive research by Douady, Hubbard and many others (Devaney, 1989).

The enthusiastic study of fractals has grown very rapidly in recent years and the ready availability of high definition computer graphics has made it possible for schoolchildren, as well as teachers, to produce a wonderful variety of exotic pictures based upon iteration of simple functions of complex variables (Peitgen et al, 1992). Even more recently the work of Barnsley on iterated function schemes, which result in remarkably lifelike pictures of ferns and trees, has aroused a lot of interest. The mathematical theory of fractals is much more demanding than their production on a computer but good progress has been made here, undoubtedly inspired by the computer graphics successes.

1.5 Algorithms

An algorithm is simply a procedure for solving a specific problem or class of problems. The notion of an algorithm has been around for over 2000 years (e.g. the Euclidean Algorithm for finding the highest common factor of two integers), but it has attracted much greater interest in recent years following the introduction of computers and their application not only in mathematics but also to problems arising in technology, automation, business, commerce, economics, the social sciences, etc. (*see* also the chapter by Maurer). Computer algorithms have been developed for many commonly occurring types of problems. In some cases several algorithms have been produced to solve the same problems, e.g. to sort a file of names into alphabetical order or to invert a matrix, and in such cases people who wish to use an algorithm will not only want to be sure that the algorithm will do what it is supposed to do, but also which of the several algorithms available is, in some sense, the "best" for their purposes. An algorithm which economizes on processor time may be extravagant in its use of storage space or vice-versa and the need to find algorithms which are optimal, or at least efficient, with respect to one or more parameters has led to the development of complexity theory. Thus the Fast Fourier Transform has reduced the time complexity of computing Fourier transforms from order n^2 to order $n \log n$, which is of considerable practical importance for large values of n. More recently the problem of designing algorithms which can be efficiently run on several processors working in parallel has attracted considerable interest.

Algorithms which are ideal on a single processor may be highly inefficient, or even fail entirely, on parallel processors. The search for algorithms for the efficient solution of mathematical problems on systems of parallel computers is a major area of research and conferences on this topic are held regularly. The problems are mathematically challenging and are also of considerable practical importance. With parallel computer systems now being readily available, courses on parallel computing are being taught at undergraduate level which, five years ago, would have been possible in very few places.

A final point is this: the growing importance of algorithms suggests an enlarged role for ***proof by algorithm*** in which a constrictive proof of an existence theorem is obtained by exhibiting an algorithm to construct the object posited.

1.6 Symbolic Mathematical Systems

The possibility of using a computer to manipulate symbols, rather than numbers, and so provide users with packages for algebraic manipulation and indefinite integration was appreciated from the earliest days of computers. Packages such as ALPAK and Slagle's SAINT (Slagle, 1963) both date from the early 1960's. Not only were such packages available, they were used, Around 1960, Lajos Tokacs used ALPAK to carry out some very tedious algebraic manipulation involving 1200 terms to find the second moment in a problem in queueing theory, of importance to Bell Laboratories. No one had had the courage or energy to do this by hand. When the second moment was finally found it reduced to just three terms, after which a shortened mathematical derivation was obtained and a general theory developed. Two points are worth noting: After the brute- force use of ALPAK the nature of the solution inspired mathematicians to find a more elegant derivation - in support of our remark in Section 1.2; secondly, without the use of a symbolic manipulation package it is unlikely that this work would have been done at all.

Another early system, FORMAC, was utilized to help with the solution of the restricted case of the 3-Body Problem and, more recently, it has been used to check that two 752-term polynomials, occurring in the theory of plane partitions, are identical.

Some symbolic manipulation packages are general, but many more are applications specific. We have mentioned CAYLEY which is widely used for the study of finite groups both at research level and as a teaching aid. Other specific systems include MATRIX, REDUCE (Fitch, 1985), MACSYMA (Pavelle and Wang, 1985); Maple (Char, 1988); Mathematica (Wolfram, 1988); many more traditional algebra systems are surveyed in Pavelle et al (1981). These are further discussed in the chapter by Hodgson and Muller.

1.7 Computers and Mathematical Communication

Whilst it affords great personal satisfaction to prove (or disprove, or conjecture) a result, the mathematical community only gains if that result is communicated to others. This communication may take various forms (though the distinctions are not rigid).

- epistolary - where A writes a letter to B communicating the result;
- proscriptive - where A writes the result on a wall (literal or metaphorical) for others to read;
- privately published - the usual form is a departmental technical report, whose existence is announced;
- publicly published - journals or books.

This communication may be received either directly by the person who is going to use the result, or indirectly.

The advent of computed-aided typesetting and camera- ready copy has obviously changed the visual form of mathematical comunication (particularly the publicly-published) and its economics. This has consequences for mathematicians (especially editors) who may need to read the input to such type-setting systems. But computer technology is capable of changing and is changing, far more than this.

Epistolary. Computer networks have revolutionized this method of communication by allowing "letters" to be sent via *electronic mail* instead of physical mail As more and more mathematicians are linked by such networks, they will replace most written communication.

Proscriptive. In addition to the physical notice boards in one's own department or elsewhere on which one can place proofs (or, more likely, announcements of technical reports containing proofs), computer networks distribute electronic "bulletin boards" to various sites which "subscribe" to them. In some areas of computer science in North America, most results are announced on such bulletin boards.

Private Publishing. This is closely related to the above. Such networks also distribute electronic "newsletters" to individual subscribers, which often contain lengthy articles in draft form, or state conjectures or problems.

Public Publishing. This is the area whose form has been directly least affected. Though there is talk of it, no serious refereed journals distributed by electronic means exist.

REFERENCES

Andrews, D.F. et al. [1972]: *Robust Estimates and Location*, Princeton, NJ: Princeton University Press.

Appel, K. and Haken, W. [1976]: The solution of the 4-color map problem, *Sci. American* (October), 108-121.

Atkin, A.O.L. and Birch, B.J. (Eds.) [1971]: *Computers in Number Theory*, San Diego: Academic Press.

Char, B.W. *et al* [1991]: *Maple V Library Reference Manual*, New York: Springer-Verlag.

Churchhouse, R.F. [1973]: Discoveries in number theory aided by computers, *Bull. IMA*, 9, 15-18.

Churchhouse, R.F. [1980]: Computer arithmetic and the failure of the associative law, *Bull. IMA*, 16, 210-214.

Churchhouse, R.F. [1985]: Computer arithmetic II: some computational anomalies and their consequences, *Bull. IMA*, 21, 70-73.

Churchhouse, R.F. and Herz, J.C. (Eds.) [1968]: *Computers in Mathematical Research*, Amsterdam: North Holland.

Devaney, R.L. [1989]: *An Introduction to Chaotic Dynamic Systems* (2nd Ed.) (A-W Studies in Nonlinearity), Reading, MA: Addison-Wesley.

Elkies, N.D. [1988]: On $A^4+B^4+C^4$, *Math. Comp.*, 51, 825-835.

Fatou, P. [1919]: Sur les equations fonctionelles, *Bull. Soc. Math. France*, 47, 161-271.

Fitch, J. P. [1985]: REDUCE, *J. Symb. Comp.*, 1, 211-227.

Ingham, A.E. [1932]: *The Distribution of Prime Numbers*, Cambridge: Cambridge University Press.

Julia, G. [1918]: Memoire sur l'iteration des fonctions rationelles, *J. Math.*, 8, 47-245.

Lander, L.J. and Parkin, T.R. [1967]: A counterexample to Eulers's sum of powers conjecture, *Math. Comp.* 21, 101-103.

Leech. J. (Ed.) [1970]: *Computational Problems in Abstract Algebra*, Oxford: Pergamon Press.

Pavelle, R. and Wang, P.S. [1985]: Macsyma from F to G, *J. Symb. Comp.*, 1, 69-100.

Peitgen, H-O., Jürgens, H. and Saupe, D. [1992]: *Fractals for the Classroom. Part One: Introduction to Fractals and Chaos*, New York: Springer-Verlag.

Ramanujan, S. [1927]: *Collected Papers*, Cambridge: Cambridge University Press.

Slagle, J.R. [1963]: A heuristic program that solves symbolic integration problems in freshman calculus, *J. ACM*, 10, 507-520.

Tukey, J.W. [1977]: *Exploratory Data Analysis*, Reading, MA: Addison-Wesley.

Wolfram, S. [1988]: *Mathematica*, Reading, MA: Addison-Wesley.

Zabusky, N.J. and Kruskal, M.D. [1965]: Interaction of "solitons" in a colisionless plasma and the recurrence of initial states, *Phys. Rev. Lett.*, 15, 240-243.

Part II

THE IMPACT OF COMPUTERS AND COMPUTER SCIENCE ON THE MATHEMATICS CURRICULUM

Anthony Ralston
SUNY at Buffalo, Buffalo, NY 14260, USA

2.0 The Changing Science of Mathematics

In this section we will consider how computers and computer science should be causing changes in the mathematics curriculum because of the changing importance of various branches of mathematics wrought by computers and computer science (see the chapter by Steen). One aspect of this change is that increasingly the knowledge of mathematics important to the user of mathematics is no longer that of detailed knowledge but rather what might be called "meta-knowledge" about the characteristics and power of methods, often numerical, for the solution of classes of problems (see the chapter by Mascarello and Winkelmann). A related perspective is that computers have brought mathematics much closer in philosophy to the classical natural sciences where there has always been an interplay between theory and experiment. Now mathematics, too, has a laboratory - the computer - on which experiments can be performed which lead to theories and on which theories can be tested. These points should be kept in mind in what follows. Although we shall not return explicitly to them, they influence much of this section.

2.1 The Common Mathematical Needs of Students in Mathematics, Science and Engineering

(a) Preparation for University Mathematics

To provide a context in which to discuss the impact of computers and computer science on curriculum and pedagogy, it is necessary to agree first, in general, on the appropriate mathematics for the secondary school student (see the chapter by Graf et al) and then to consider the university curriculum. Since there are significant differences between different parts of the world on when secondary school ends and university instruction begins, the comments which follow will have to be interpreted in the local context.

Algebra has traditionally been an important subject in high school. Since elements of abstract algebra are likely to become increasingly important in mathematics education, it is clear that algebra will remain of central importance in the secondary school curriculum. The important thing, however, is not to have students achieve great manipulative skill in algebra (e.g. in polynomial algebra) but rather to teach them to consider algebra as a natural tool for solving problems in many situations. Nevertheless, the ability to use formulas and other algebraic expressions will remain necessary.

In recent years there has been a trend toward replacing much of Euclidean plane geometry with those aspects of geometry more closely akin to algebra. This is useful as a preparation for university mathematics but there is much feeling among mathematics educators that the loss of Euclidean geometry is a sad development. A consensus on how geometry might best be taught at school and university is not yet available. It should be noted, however, that some computer scientists feel that the aspect of traditional instruction in geometry concerned with teaching the meaning and construction of rigorous proofs can be achieved through material concerned with the analysis and verification of algorithms (see the chapter by Maurer).

For many parts of mathematics trigonometry is useful preparation. But we note that much of the tedious work which was necessary in the past, both numerical and symbolic, can now be easily done on hand-held computers.

Next we mention calculus. In many countries this has been a secondary school subject for many years for most university-bound students while in other countries only the very best students begin calculus in secondary school. The main thrust of secondary school calculus has been to provide students with techniques, and to prepare those intending to study mathematics at university with a first introduction to the concepts they will encounter at the university level. Since all the techniques of secondary school calculus as well as much of university calculus can now be done on hand-held devices or on symbolic mathematical systems (often called "computer algebra" systems) on computers (see the chapter by Hodgson and Muller), calculus at the secondary as well as university level must focus almost entirely on concepts and not on computation.

Various new subjects have become part of the secondary school curriculum in recent years. Among these is probability which has come into the curriculum in many countries. Topics such as discrete probability distributions, the binomial distribution and related topics are appropriate. So too, is an introduction to data analysis and elementary statistics because of their importance in science as well as mathematics. Another subject, about which there will be further discussion below, which we would like to see more of in the secondary school curriculum, is discrete mathematics including elementary combinatorics and graph theory as well as an introduction to induction and recursion. In this connection it would be appropriate to introduce both the design and verification of a number of important algorithms such as those for sorting. Finally we note that elementary linear algebra, particularly matrix algebra and work with systems of linear equations, should certainly be considered for the secondary school curriculum.

(b) The University Mathematics Curriculum

The core of the university mathematics curriculum for many years has been the calculus and, to a lesser extent, linear algebra. This is the case no matter how much mathematics the student may have studied in secondary school. Computers themselves have an impact on both the content of this curriculum and its pedagogy. Not only do computers allow more interesting and effective presentation of classical subject matter but, in addition, as with the secondary curriculum, they affect what subject matter is important to students. For example, symbolic mathematical systems suggest a deemphasis on the more skill-oriented portions of the current curriculum.

Informatics (i.e. computer science) itself also implies changes in the content of the core curriculum. This is essentially because informatics is a highly mathematical discipline whose problems require almost universally the tools of discrete rather than continuous mathematics. Thus, there is now a strong argument to provide a balance in the core curriculum between the traditional continuous mathematics topics and topics in discrete mathematics (Ralston 1981, Ralston and Young 1983, Ralston 1989). For university courses aimed at a broad spectrum of mathematics, science and engineering students, this balance may well contain nearly equal portions of the continuous and the discrete. For those courses aimed at specific student populations, the balance might be weighted more in the direction of the discrete for informatics and social and management science students, might be about equal for mathematics students themselves and surely should be weighted more toward traditional continuous mathematics for physical science and engineering students. It needs to be emphasized, however, that all groups of students need some exposure to both the continuous and discrete approaches to mathematics. Whether students are exposed to calculus first and then discrete mathematics or vice versa will depend on the student population and on institutional convenience.

The actual content of the discrete mathematics component is still quite variable. However, the discrete component normally contains at least some "traditional" discrete mathematics (e.g. combinatorics, graph theory, discrete probability, difference equations) as well as perhaps some abstract algebra although the latter may follow in a later course after completion of the core courses.

We note also the importance of mathematical logic in the core university curriculum. Although traditionally an advanced undergraduate or a postgraduate subject (at which levels there will be a continuing need for specialized courses), logic is so important in informatics that it needs to be introduced early in the university mathematics curriculum. Moreover, with the increasing need for people in the scientific and technical professions to handle information in a precise manner, logic has great value for a wide variety of students. Logic is an important constituent of many discrete mathematics courses (see below). But it can also be considered as a subject for a course by itself which would follow the introduction in discrete mathematics. Such a course can usefully be given a distinctive computer flavor as described by Schagrin et al (1985).

As a final matter, we stress the importance of using the paradigms of informatics (e.g. an algorithmic approach, iteration, recursion) in the teaching of mathematics at all levels. Although these paradigms may seem most easily applicable to discrete mathematics, there is considerable scope for their introduction into the classical continuous curriculum.

The reader may be surprised to find no mention of numerical analysis here (or hereafter in this document) because this subject is the one that most obviously combines the continuous and discrete approaches to mathematics. But we take the position that numerical analysis is now such a well-established subject in the mathematics curriculum that it does not need to be discussed in the context of this report. This is, however, not to say that the subject matter of numerical analysis is no longer affected by advances in computing; developments in, for example, parallel computing are having great impact on numerical analysis.

2.2 A Discussion of Particular Curriculum Areas on Which Computers and Informatics Have an Impact

Although discrete mathematics and calculus are discussed separately in what follows, it should be emphasized that there is no intellectual reason to consider them as separate subjects. Indeed, they are mutually supportive and ideally would be taught in integrated courses (see the chapter by Seidman and Rice). However, for at least some years to come, such integrated courses will be relatively rare, not least because of the lack of textbooks for integrated courses.

(a) Discrete Mathematics Courses

We begin with a discussion of what topics in discrete mathematics should be contained in courses intended for mathematics students as well as for students in the social and management sciences. Although the topics to be listed below cover a broad spectrum, it is possible to design a coherent course covering these topics if the course is built around themes such as algorithms and their analysis and inductive and recursive thinking.

A Discrete Mathematics Syllabus

1. Mathematical Preliminaries - Sets, functions, relations, summation and product notation, matrix algebra, an introduction to proof and logic concepts.
2. Mathematical induction including its application to algorithms and recursive definitions.
3. Graphs, digraphs and trees including path, searching and coloring algorithms, tree traversal, game trees and spanning trees and applications in a variety of areas.
4. Basic Combinatorics including the sum and product rules, permutations, combinations and binomial coefficients, inclusion-exclusion, the pigeonhole principle and combinatorial algorithms.
5. Difference equations (i.e. recurrence relations) including first order equations, constant coefficient equations and the relationship of recurrence relations to the analysis of algorithms, particularly divide-and-conquer algorithms.
6. Discrete probability including random variables, discrete distributions and expected value.
7. Mathematical logic including the propositional calculus, Boolean algebra, the verification of algorithms and an introduction to the predicate calculus.
8. Infinite processes in discrete mathematics: Sequences, series, generating functions, approximation algorithms.

In addition, other possible topics depending upon local needs and desires are:

9. Algorithmic linear algebra including the use of Gaussian elimination as an entree to abstract linear algebra and an introduction to linear programming and applications of linear algebra.
10. Decision mathematics including such things as queueing theory and packing problems.
11. Algebraic structures such as rings, groups etc.
12. Finite state machines and their relation to languages and algorithms.

And, of course, there can be extensions of all the above topic areas to more advanced subject matter if desired and appropriate.

Since the Strasbourg conference in 1985, at least 40 books have been published from which a course on the above lines can be taught (see, for example, Epp, 1990 and Maurer and Ralston, 1991).

The experience of those who have taught such courses is that, despite the potpourri of topics listed above, these courses can be made interesting and satisfying if a consistent, coherent approach is taken which emphasizes algorithmic, recursive and inductive thinking.

Following a course from a syllabus like that above, a variety of advanced courses in discrete mathematics can be contemplated although only the largest institutions would be able to offer all of these. Indeed, each of the subject areas listed above suggests one or more advanced courses which would build on the introductory material in a first discrete mathematics course. Most of these courses are currently in a process of evolution as the subject matter in the first discrete mathematics course changes and develops and as the applications of discrete mathematics grow and diversify. A program which combines a carefully constructed introductory discrete mathematics course with several advanced courses will give the student a firm basis for studying informatics as well as providing a basis for professional work in modern applied mathematics and other fields in science and engineering.

(b) Calculus in the Computer Age

(i) The Role and Relevance of Calculus

Among the key factors which compel change in the teaching of university mathematics courses are:

- the substantial experience with minicomputers and microcomputers and programming packages which many students have had before coming to the university;

- the growth of new areas of applied mathematics such as the analysis of algorithms and computational complexity.

One result of this is that many students have attitudes and expectations which lead them to believe that the most challenging and meaningful mathematical problems today are related to computers and informatics. This cannot help but influence how we must motivate mathematics students and all other students in mathematics courses.

In considering the place of calculus in the computer age, we cannot forget that it is one of humankind's great intellectual achievements. Every educated person should be aware of it. Its history exemplifies the "unreasonable effectiveness" of mathematics better than any other branch of mathematics. And its effectiveness is as great today as it has ever been. But this does not excuse teaching calculus as is so often the case now with an emphasis only on the execution of mechanical procedures - and paper-and-pencil procedures at that. Instead calculus needs to be taught to illustrate the unique ways of thinking it epitomizes.

The realm of applications of calculus remains immense. They are, indeed, increasing due to the increasing mathematization of heretofore qualitative sciences like biology. In constructing calculus models of phenomena and then solving the resulting equations, there is often an interplay between these models and their discrete counterparts with the calculus models representing the limiting behaviour of the discrete models. It is now more important than ever to include this interplay in calculus (and discrete mathematics) courses because inevitably the solution of most problems in calculus involves the (discrete) computer. The discretization necessary to solve problems of calculus with a computer often has not borne a close relationship to the underlying discrete model. But the increasing power of computers means that more and more frequently it is possible to have computer models which mirror very closely the discrete models from which the continuous model was initially abstracted.

There already are powerful software tools which can be used in the study of calculus. These include symbolic mathematical systems and a variety of graphical packages. Advances have taken place so rapidly in these areas that it is now the case that very powerful symbolic and graphical systems are available on hand-held computers (e.g. the HP-48S) as well as on microcomputers. One result of this is that an understanding of functions, variables, parameters, derivatives etc. and the ability to interpret formulas and graphics is becoming more important to the student than skills in executing the (numerical or symbolic) procedures of calculus. In the teaching of calculus to all students the need is clear for a shift from an emphasis on calculational technique to one which emphasizes the development of mathematical insight.

(ii) The Content of Calculus Courses

If functional behaviour and representation are to be the focus of the calculus course, then continuous functions and discrete functions (i.e. sequences) must be emphasized and motivated by a wide variety of mathematical models. (Indeed, mathematical models and their applications in a variety of disciplines should be an important part of calculus courses.) (Note: it can be argued that sequences belong more properly in the discrete mathematics course discussed previously. This only illustrates the need to bring the discrete and continuous points of view together into an integrated sequence of courses as soon as possible.

An important theme in calculus courses should be the contrast between the local and global behaviour of functions. Local behaviour is, of course, derived by studying the derivative for continuous functions (and the difference operator for discrete functions). And similarly the integral (and summation) operators are used to derive global information about functions. Undoubtedly it will remain necessary to develop some ability to do formal computations with derivatives and integrals. But the major emphasis should be on numerical algorithms (particularly for integrals) and on how derivatives and integrals can be used to understand the behaviour of functions.

A topic such as the Taylor series representation of a function should be used to show how good local information can be obtained using low-degree Taylor polynomials and interpolating polynomials, another area where the analogy between the continuous and the discrete may be usefully shown.

Finally, there should be a balance in the calculus course between traditional topics and ones whose importance has greatly increased because of the advent of computers and informatics. Thus, for example, the $O()$ and $o()$ notations, which enable the asymptotic growth rates of functions to be compared to standard functions like polynomials and logarithms, are not always taught in calculus courses, but they should become so.

This discussion is intended only to provide the flavor of how an orientation toward computation should change the approach toward teaching most of the standard calculus topics.

(iii) Computers for Learning and Teaching Calculus

Computers enable teachers to modify their methods of teaching calculus (and, of course, much other mathematics also) in order to meet better the need of their students. Computer graphics is a powerful medium in which to provide examples - and non-examples - of continuous functions, discontinuous functions, the area under a curve, direction fields and nowhere differentiable functions as well as in many other areas. Well-designed software (there still isn't nearly enough of this) can be used by students to discover and explore the concepts mentioned above as well as such fundamental concepts as slope and tangency (see also Section 2.3). But the effective use of such software requires that teachers sometimes depart from a lecturing style and go instead to guiding and interacting style with small groups of students or individual students.

Well-designed software will also permit enhancements by students through the writing of (usually short) programs. This is just another way in which students can be actively involved in their own learning although it is important that the use of the computer does not become the message instead of the mathematics which it is supposed to illustrate. Thus, programming per se should not play any significant role in a calculus course.

Another impact of the computer in calculus may be to change the order in which topics are taught. For example, it is becoming increasingly common to introduce limits at the very start of a calculus course. Tangent functions and area under a curve can be motivated and defined graphically. When a formal definition of a limit is needed, students will be ready for it. As another example, differential equations can now be treated much earlier in the curriculum than was previously possible because of the ease of understanding made possible by new graphics systems (see the chapter by Tall and West). They can be introduced right after differentiation and before integration. Studies are now under way to discover whether such reorderings will lead to a greater or more rapid understanding of fundamental concepts and theorems.

To take full advantage of the use of computers in teaching calculus, it will be necessary to change the standard classroom environment. Classrooms need to be provided with large monitors or screens on which the monitors may be projected. Both inside and outside the classroom, students need administratively easy and user-friendly access to computers and software. Teachers will need private computer facilities in order to prepare course material. A prerequisite for this is in-service training so that teachers may become comfortable with computers and then fluent in their use and aware of possibilities beyond what may be available in the particular software on which they have learned.

Finally, we note the value of using computers in the classroom to teach mathematics. The desirability of this for calculus and related subjects is particularly clear since the dynamics of computer graphics is ideally suited to help explain a subject which is essentially about change. Indeed, it is ironic that only static technology - the chalkboard and the overhead projector - are still used so widely to teach calculus. There is a considerable amount of software available now which can be used in the classroom to teach calculus (e.g. Flanders (1991)) and differential equations. There is much less software available to teach discrete mathematics in the classroom but there are numerous aspects of discrete mathematics (e.g. induction and recursion) for which suitable software would be valuable in the classroom. We can expect to see the development of such software in the near future.

2.3 Exploration and Discovery in Mathematics

The idea of using computers to enable students to explore mathematics and discover mathematical patterns for themselves is not a new idea (Steen, 1988). However, the advent of powerful and available computer systems makes this point so important in teaching mathematics today that we devote an entire section to it.

First, why should exploration and discovery be important components of the educational process in mathematics? The answers parallel the reasons why we teach mathematics in the first place:

- active learning leads to better retention and understanding and more liking of the mathematics we teach because the mathematics is seen as a basic component of human culture; it also leads to more self-confidence in the ability to use mathematics to solve problems;

- exploration and discovery helps to teach people to think;
- discovery provides the greatest aesthetic experience in mathematics, the "aha" of seeing or proving something is what makes mathematics attractive;
- exploration and discovery are perhaps the best ways for students to see that mathematics is so useful;
- discovery enables the student to see a familiar idea applicable in a new context, thereby enabling a grasp of the power and universality of mathematics.

Computer technology may be used to assist in mathematical exploration and discovery in a variety of ways; for example:

- through visualization of a great variety of two and three dimensional objects via computer graphics, students may explore questions and discover results by themselves.
- through computer graphical presentations of interesting geometries like "flatland" and turtle geometry;
- via exploratory data analysis to, for example, draw conclusions from data (e.g. is it bimodal? are there outliers?), to transform data (e.g. by logarithmic plots), to smooth data and to compare different sets of data.
- by graphical and numerical explorations of how to approximate complicated functions by simple ones;
- by applying the first step of the inductive paradigm - compute, conjecture, prove - in many, many different situations;
- by using symbolic mathematical systems to discover mathematical formulas such as the binomial theorem;
- by designing and executing different algorithms for the same or related tasks.

This list could be made much longer. Readers will probably be led to make their own suggestions.

There are various implications to using computers to facilitate exploration and discovery:

- we must start with easy tasks so that students feel they are really succeeding on their own and are not being led step by step by the teacher;
- teachers need to be educated for this kind of instructional mode; few teachers can become comfortable with these ideas without explicit education; we note, in particular, that testing what has been learned by the student is not easy. But experience has shown that success is not only possible but yields rich rewards. The difficulties can be overcome; teachers can be trained to feel comfortable with this mode of learning.

2.4 Some Speculation about the Future

As mathematics becomes increasingly an experimental science, it is inevitable that computers and computer science will have increasing influence on the mathematics curriculum. Computer science will become a gradually greater focus of applications of mathematics and this will affect what is important in mathematics. At the same time the means by which all mathematics is taught will be inextricably entwined with computer technology. Although the cost of this technology will continue to be a problem for developing countries, the curricular inertia in developing countries is far less than that in the developed countries. Developing countries have an unparalleled opportunity to use computers and the influence of computer science to modernize their mathematics curricula and their mathematics teaching faster than will be possible in developed countries.

REFERENCES

Douglas, R.G. (Ed.) [1986]: *Toward a Lean and Lively Calculus*, MAA Notes Number 6, Washington, DC: Mathematical Association of America.

Epp, S.S. [1990]: *Discrete Mathematics with Applications*, Belmont, CA: Wadsworth.

Flanders, H. [1991]: *Microcalc 6.0*, Calculus Software for VGA, Ann Arbor, MI.

Maurer, S.B. and Ralston, A. [1991]: *Discrete Algorithmic Mathematics*, Reading, MA: Addison-Wesley.

Ralston, A. [1981]: Computer science, mathematics and the undergraduate curricula in both, *Amer. Math. Monthly*, 88, 472-485.

Ralston, A. (Ed.) [1989]: *Discrete mathematics in the First Two Years*, MAA Notes Number 15, Washington, DC: Mathematical Association of America.

Ralston, A. and Young, G.S. [1983]: *The Future of College Mathematics*, New York: Springer-Verlag.

Schagrin, M.L., Rapaport, W.J. and Dipert, R.R. [1985]: *Logic: A Computer Approach*, New York: McGraw-Hill.

Steen, L.A. [1988]: The Science of Patterns, *Science*, 240 (29 April), 611-616.

Part III

COMPUTERS AS AN AID TO TEACHING AND LEARNING MATHEMATICS

B. Cornu
IUFM, Grenoble, France

3.0 Introduction

Mathematicians and mathematics teachers have been provided with a new tool, the computer. There is no shortage of applications or interesting examples which one can quote. But, like all tools, the computer by itself does not supply a solution to the problems of mathematics education. There is no automatic beneficial effect linked to a computer: The mere provision of micros in a class - or lecture room will not solve teaching problems. It is essential, therefore, that we should develop a serious programme of research, experimentation and reflective criticism into the use of informatics and the computer as an aid to teaching mathematics. It will not suffice to think only in terms of mathematics and the computer, and of the production of software which amuses and interests mathematicians. We must also take into account types of knowledge and the ways in which these can be transmitted, and attempt to study, in a serious epistemologically-based manner, various concepts and the obstacles which they present to learners. We must think of students, their development and the matching of new and old knowledge. We must consider in depth the teaching possibilities created by the computer. It is essential, above all, that we should move beyond the stage of opinions, enthusiasms, and wishful thinking and engage in a true analysis of the issues. Only in this way will we come to a true resolution of certain problems of teaching. Such research, of necessity experimental, will have to be critically evaluated. It must be shown how, in given circumstances, the use of the computer can facilitate the acquisition of a particular concept. Finally, such research work will have to be built upon and developed to provide a vital component in the training (whether formal or self-directed) of teachers and lecturers. Only then can computers have any large-scale effect on mathematics teaching.

Certainly such research has been done in the past few years, and we can now see examples of uses of computers in education, based on a serious study of the didactical problems to be solved. In such uses, the computer is not a tool supplementary to traditional teaching; it is integrated in a pedagogical strategy, adapted to the actual obstacles the students have in learning. But much remains to be done. Both the development of educational research and the evolution of technology have the potential to effect major changes in teaching and learning in the future.

Computers for mathematics teaching are not so widely used as one could think. Appropriate software and strategies do exist; but they are used by few teachers; one of the main problems now is to help all teachers to use computers, not as a new experience, but as a common tool for teaching. This requires not only good training for teachers and good pedagogical products and tools, but also good integration of new technologies in curricula and good long term pedagogical strategies.

3.1 A changing view of mathematics

There are many references in this book to the way in which the computer can lead to a changed view of what mathematics and mathematical activities comprise. For example, as the experimental aspects of mathematics assume greater prominence (see Section 2.3), and there is a corresponding wish to ensure that provision should be made for students to acquire skills in, and experience of, observing, exploring, forming insights and intuitions, making predictions, testing hypotheses, conducting trials, controlling variables, simulating, etc. Examples of how such work can be carried out are found in later chapters in this book. However, mechanisms need to be found for disseminating information about fruitful experimental environments and how these can be formed.

Yet, as we put new emphasis on the particular activities listed above, it is also necessary to ensure that such traditional activities as proving, generalising and abstracting are not neglected or omitted. We will need to find an appropriate balance between 'experimental' and more formal mathematics.

The possibilities presented by the computer will actually help focus our attention on the kind and types of knowledge which we wish students to acquire. Not only are new possibilities offered to us, but also a greater incentive to identify more precisely our educational goals.

If our aims of teaching change significantly so as to encompass and stress the 'process' of mathematics more than the 'products' of the mathematical activities of others, then there will, of course, be a need to identify those parts of mathematics most suitable for our purposes. Topics and areas of mathematics must be selected which encourage and facilitate an experimental approach.

Finally, in this section we must stress two important, interrelated points. Many, indeed the majority, of our students do not intend to become mathematicians. We must not lose sight of the implications of this in terms of educational goals and emphases. But, also, many of these may be students of the experimental sciences. This raises further important issues, for experiments in mathematics differ somewhat from those in the physical and natural sciences. The techniques are often very similar, but in mathematics we have that extra, vital ingredient of 'proof'. Experiments are an essential and neglected part of mathematics, yet mathematics is not an experimental science. The distinctions between disciplines and ways of thought will have to be displayed and observed.

3.2 Computers change the relation between teacher and student

Computers can affect the behaviour of students. This creates new interactions and relationships between student, knowledge, computer and teacher. The role of the teacher in such situations demands considerable thought.

(a) The mathematical activity of the student

Students will be better able to learn conceptual material and develop autonomous (as opposed to imitative) behaviour patterns with respect to mathematical ideas if they can be cognitively active in response to mathematical phenomena presented to them. This activity should consist of the formation of mental images to represent mathematical objects and processes. It should also include the development of skills in manipulating these objects and processes. In this way students can increase their ability to think mathematically.

Inducing students to emerge from passivity and to think actively about mathematics is, however, not easy. One approach is to make use of the computer to supply sufficiently powerful and novel experiences to stimulate such behaviour. The action of a computer program and the structure of data as it is represented in the computer can form useful models for thinking about mathematical entities. For example, a "WHILE loop" whose body is a simple sum is a process that can represent the mathematical entity

$$\sum_{i=1}^{m} x_i.$$

This expression, which troubles so many students, can then be thought of in terms of a simple, familiar and useful computer process. Again, in Pascal, representing a fraction as a record with two integer fields (the second being non-zero) helps students think about rational numbers as ordered pairs of integers, especially if they are given the experience of writing programs to implement the arithmetic of fractions without truncation.

More generally, many mathematical concepts can be defined or described as procedures. This gives a more dynamic approach, and can help the student in understanding and in using these concepts. Algorithmics (see the chanpter by Maurer) gives many tools for introducing mathematical concepts in such a way.

Many examples of ways in which such experiences can be incorporated into mainstream, tertiary-level courses are available. Moreover, the success of such initiatives would seem to be independent of several issues which in discussion tend to be overrated. An important factor in this approach appears to be that students should write the programs and so *must* be cognitively active about the processes and data structures they are implementing. These experiences are then coordinated with classroom activity.

In their chapter, Mascarello and Winkelmann describe a course containing 'continuous' topics such as multiple integration and ordinary differential equations. Here the students wrote programs in a low-level language running on a microcomputer. These were interactive and the results were used for experimentation and demonstration.

Of course, writing programs is not the only useful way in which students can use the computer. The use of complicated software packages for illustration of phenomena that are very difficult to display otherwise can clearly broaden the students' awareness and add to their general understanding (see, for example, the chapter by Tall and West). They can, of course, also be used for exploration and discovery. Indeed, some would see the most exciting opportunity offered by the computer to be the way in which it can motivate students to exercise the process of discovery. Here we should only stress the need to see exploration and discovery as essential mathematical activities to be practised. Traditionally, this has not been so - teaching and learning have been almost wholly concerned with the transmission and reception of accepted mathematical facts. However, now, for example, computer symbolic mathematical systems (see the chapter by Hodgson and Muller) permit such rapid and flawless processing of non- trivial examples that it is easy first to look for patterns which suggest conjectures and generalisations, and then to search for counterexamples or machine-aided proofs.

Computers then can greatly assist us in extending the range and the depth of students' mathematical activities. In some approaches the students will write their own programs (and there will be an attendant risk that mathematical aims may become obscured by some of the programming problems); in others students will use prepared software. Both approaches have already been shown to be of great value; further investigations into both will now have to be continued.

(b) The role of the teacher

The computer can be used in two distinct ways in the classroom. In one it is an aid for the teacher, an electronic blackboard – more powerful than the traditional blackboard, the overhead projector, or a calculating machine – but nevertheless a tool whose output is almost entirely under the teacher's control. In this role the computer does not upset the traditional balance in the classroom. It will still demand effort on the teacher's part to select or provide suitable software and it can give rise to extra administrative problems; in return it should enhance learning. However, it will not revolutionise the classroom.

If, however, students are allowed and expected to interact with computers, then the position changes, for this leads of necessity to a change of methodology. The teacher no longer has total control – his/her role can no longer be limited to exposition, task-setting and marking. The format 'lecture-examples, homework-exam' must be augmented by, for example, 'project (through interaction between student, machine and teacher) assessment on the basis of a completed (and possibly debugged) assignment'.

Probably the teacher must combine diverse uses of the computer. Some activities fit well with the 'blackboard computer'; some others will be more efficient if each student has the opportunity to interact with the computer.

Such a change would produce a revolution in most class- and lecture-rooms. It demands that teachers should not only acquire new knowledge, skills and confidence in the use of hardware and software, but that they should also radically change their present aims and emphases, and accept a lessening in the degree of control which they presently exert over what happens in their classrooms. This last demand means a sacrifice of traditional security, at a time when teachers will still be fighting hard to gain new skills and acquire confidence in them. It would be foolish to underestimate the challenge this presents.

The acquisition of new skills will be time-consuming and constantly changing hardware and software will make the process a continuing one. For many mathematicians these new skills will be readily usable in their research work. Others may be tempted – particularly when universities and other educational institutions are under pressure – to feel that such time would be more profitably spent in increasing personal research output, rather than in improving their teaching, particularly if this requires such a large step in the dark.

Computer usage is still actively avoided by many mathematicians and by many mathematics teachers. The problem at the tertiary level is particularly great, for the gulf between the traditional lecture often given to a hundred or more students and the classroom/laboratory in which students interact with computers is enormous. To bridge this gulf will need considerable investment in both material and human resources. Time, assistance and in-service training will have to be provided on a scale unprecedented at this level. Particular attention will have to be directed at those teachers who still have many years - even decades - to go before they retire from teaching. First, however, the necessity for change will have to be accepted, and this will only come through clear, unequivocal demonstration of the benefits which can accrue from innovation.

The current problem now is to make **all** teachers able to use computers in teaching, or to know why they will not use them! This leads to different problems:

- The availability of computers in the teachers' environment: Can they easily find and use a computer at home for preparing their teaching and elaborating activities? Can they easily find and use a computer in the school? Are computers easily available in classrooms?
- The user-friendliness of hardware and software: Will it take hours and hours to prepare a lesson with computers, and will very specialised abilities to use such software be needed?
- The integration of the computer in the teaching strategy and in the learning environment. The computer is a tool among others, and its use must be integrated in a pedagogical strategy. Textbooks, homework and all the activities of the learner must take this into account.

The computer does not only change the teacher's role, but also the attitude and activities of the students. The relationship is not only between the teacher and the learner: The computer takes its place in the relation, and it also develops the group work and the project activities. Learning from pupil-pupil talk is one of the components of the learning environments provided by new technologies. Different types of environments, different teaching methods and different strategies can be used by teachers. Computers change the organisation of education, and give teachers the role of a pedagogical-engineer in elaborating their strategy, in preparing their teaching, in choosing among the resources available and the tools and products they will use.

3.3 Some particular uses of the computer in the classroom

We have already remarked on the way in which computers can assist in the introduction, development and reinforcement of mathematical concepts, in building up intuition and insight, etc. In this section we look at particular ways in which they can be used within the classroom.

(a) Graphic possibilities

Many of the applications of computers in teaching make use of the possibilities provided for graphic display. There can be no doubt about their value in providing quickly good quality graphic illustrations which can help build intuition. The example of ordinary differential equations such as $x' = x^2 - t$, whose solutions cannot be written down in elementary terms, is now widely known and used: Visualising the field of tangents and visualising many solutions of the equation make the student better able to understand the concepts which intervene in this domain. Moreover, this allows them to discuss exciting questions concerning the behaviour of solutions.

Where the computer scores over many other media is that graphics capabilities now enable movement as well as static diagrams to be portrayed. This, of course, was true of the film. Yet now the possibility of being able to change parameters adds a completely new dimension to the teaching/learning experience.

Much interesting and high quality graphic software is now available and allows visual representations from areas such as calculus, differential equations, linear algebra, numerical analysis, and geometry.

A famous example in geometry is that of Cabri-Geometré which allows pupils to draw geometrical figures very easily, and to modify them by moving some elements (points, lines, etc.), and see at the same time how the figure evolves. Invariants and loci can be visualised in a very user-friendly way. Such software can be used by the teacher for demonstrating or by the students in an interactive way.

(b) Many types of **utilities** are available for use in teaching. **Spreadsheets** are the best known example. They provide a good environment for introducing many concepts in arithmetic, algebra, and even calculus. At a very elementary level, they permit interesting activities about the concepts of variables, parameters, unknowns, etc. They also provide nice illustrations of iteration. They are increasingly used in teaching.

(c) Databases are now more easily accessible. They suggest documentation activities, they allow students to look for sophisticated information and so develop project work. They also give teachers the possibility to use or build large sets of exercises and activities. The distant interrogation of databases is now common and enlarges the resources for teachers.

(d) Artificial intelligence and problem-solving tools are developing. The first step is to have software and tools able to solve mathematical problems. The second step is to produce software able to help students in problem solving.

(e) Hypermedia and multimedia products: These allow the integration of different media, and their combination for educational uses. They allow activities which are not 'linear', but in which users may build their own paths and organise their own learning. They considerably enrich other educational tools, linking and making simultaneously available all existing types of software and other educational technology products. We surmise that in the future this domain will provoke great changes in the use of technology in education.

(f) Self-evaluation and individualised instruction

One of the advantages of the computer is that it helps the individualisation of teaching and learning. This is not only because the student can sometimes work alone with the computer, but mostly because the computer can help to provide a teaching environment which matches the needs of each student – the way he learns, the right speed for her, the appropriate activities.

The computer can provide a tool for self-evaluation and can help students to take charge of the organisation of their own work. It is a difficult problem for students to judge how well they are coping with a subject. One use of computers is to enable students to test themselves. Question banks can be made available and instantaneous scores given.

The advantages of Computer-assisted Learning (CAL) for individualised instruction have, of course, been argued for some twenty years, namely that the computer can offer non-threatening, individualised responses to students. There have, indeed, been several demonstrations of the value of CAL, for example, PLATO in the USA. However, as the cognitive complexity of what has to be learned increases, the difficulties of producing adequate software become very great.

The problems become less pronounced when the aim of the program is to revise or to exercise and not to teach. Thus 'Recalling Algebra' and 'Recalling Mathematics' (Kinch) are examples of software designed to help students prepare for the Entry Level Mathematics Exam at California State University which have been favourably received.

More and more, educational software includes a "counsellor", helping the student to make his or her way through the activities of the software, evaluating him or her, and individualising the activities.

(g) Assessment and Recording

The computer can be used for testing students' progress. Some software employs the random generation of test items. Such testing can, of course, go far beyond reliance on multiple choice items and can measure responses other than correct and incorrect. Such newer testing procedures, which can be designed to capitalise on the graphic potentialities of the computer, can reduce testing time, allow tests to be broken off and resumed at any time, offer immediate summaries and analyses, and assign specific help for identified deficiencies.

The obvious disadvantages include preparation costs and the need to provide ready access to a computer. Open-ended testing of projects or personal problem solving is at present difficult, but beginnings are being made. Computer-assisted recording also has great potential.

A computer at home, or a computer easily usable at school, enables students to use individualised sets of data for homework or assessments

Very interesting examples of 'learning credit cards' are being experimented with: The card contains information about the learner, and gives him or her access to appropriate software and activities.

(h) Pocket calculators must be mentionned here. Even if their possibilities are small in comparison with computers, they are improving very rapidly: We now have calculators with graphic possibilities and even with symbolic capabilities. And the permanent availability of pocket calculators gives them great power. In many countries, the use of pocket calculators in mathematics has been introduced into the curricula so that all teachers and all pupils use calculators.

(i) Student errors

Related to the possibilities described above is that of investigating the errors which students make in learning mathematics. Such information can be used in two ways: To help the student remove misconceptions, which is its role in individualised CAL, or to help the mathematics educator to identify specific points of difficulty and to design curricula with these in mind. Errors are symptoms which allow us not only to identify stumbling blocks, but also to form an impression of the student's *conceptions*. The computer allows students to respond to their errors in a new way: They can identify and control them themselves. Getting rid of them can even become a motivation for learning.

One example of the use of the computer to detect and correct errors is found in Okon-Rinné's courseware. This enables a student to choose a basic function such as $f(x) = |x|$ and then to experiment with the effects which translations and reflections have on it. Thus the graph can be translated vertically or horizontally or reflected in the vertical axis. Simultaneously the function changes to correspond to the new graph. The intention is to detect such common errors as confusing $f(x) = |x-2|$ with $f(x) = |x+2|$, or $f(x) = |x+2|$ with $f(x) = |x|+2$. When an error is detected, a tutorial subroutine is activated and afterwards the student has the option of continuing or branching back to an earlier unit.

3.4 Student responses to work with computers

It is common to talk about the enthusiasm generated in students by computer-based systems. In many experiments, it is claimed that this has resulted in students developing a new interest in the subject and that the general level of student activity had increased as a result of reacting with a computer package. Not only had activity increased, but so had confidence. Dubinsky typically reports (of a course on discrete structures): 'This approach makes for a lively course in which students are responsive in class and active outside class. In comparison with similar groups to whom I have tried to teach this material, these students seem to be more prone to speak in terms of sets and less confused by complicated logical statements'.

It must not be thought, however, that enthusiasm can be automatically generated through the use of a computer. Much will depend on the students and the teaching situation; there are also negative experiences to report! One must also judge on how much students learn as well as the enthusiasm they show whilst engaged on the task.

Here one is faced with a new problem in teaching. Students can frequently appear fascinated by computer demonstrations or by working interactively with a computer, but what happens 'when the machine is switched off'? Will the students only be able to imitate what they have seen or will they obtain a deeper understanding of concepts?

It is recognised that the value of much computer work is largely dependent upon the follow-up activities which 'must guard against the possibility that the machine is doing all the work and providing all the answers'. Many traditional activities will still have to be carried out, thus suggesting yet again that the computer's main contribution will be to enhance student understanding and not to save time for the lecturer. The introduction of the computer is unlikely to solve (or even ease) the problem of overloaded syllabuses.

3.5 The provision of software

The way software is conceived and designed evolves very quickly. The progress of technology and the development of multimedia tools enriches the possibilities for pedagogical uses. The roles of teachers, of pedagogs and of computer scientists in software design has evolved. Very user-friendly systems allow any teacher to create teaching situations with computers.

Current software resources may be considered in several categories:

(a) Sophisticated systems (in computer terms) such as the symbolic manipulation systems, large statistical packages, etc., form the first category. These systems have been developed in a 'goal-oriented' fashion, that is they seek to provide solutions to specific mathematical problems. They have not needed to consider to any great extent 'pedagogical design'. Interest in their use as pedagogical tools is growing.

Commercial companies exist with an interest in marketing this type of software and research mathematicians are involved in creating such systems. As a result, sophisticated packages are self-perpetuating, Since they will exist, we need to understand their pedagogical uses and the possibly dramatic effects they could have on current mathematics education.

(b) Less sophisticated in computer terms but still very demanding in pedagogical design are the software packages suitable for use on a microcomputer. These packages attempt to aid the student's mathematical development and employ such themes as visualisation, simulation, exploration and problem-solving. They may be used by students working alone, in groups, or with a teacher. Many individuals and groups are writing such packages. Many are also provided by educational software companies.

A major problem arises here. The production of packages that can be recommended for widespread use as pedagogically sound and well-tested is an expensive, complicated task requiring considerable professional resources. It should involve ***fundamental research*** based on the structured observation of the materials in use in parallel with the development of the materials. Thus the team may need to include mathematicians, educators, psychologists, computer scientists, graphic designers, publishers and editors. The financial needs of such a group would be considerable.

(c) General purpose programming languages can be used as tools aiding students' mathematical development and are a readily available teaching resource. Extension of such languages or even creation of new ones expressly for this purpose would be welcome.

This brief discussion of the present position points out the need (i) to establish channels of communication so that researchers and educators are aware of resources currently available and (ii) to establish structured research studies using currently available resources in order to gain and share understanding of their use as pedagogical tools.

The emergence of software packages has raised a new problem for mathematics teachers, that of black boxes, for they often/usually produce answers without giving any hint of the way in which they were obtained. This may well conceal a wealth of deep mathematics. (It could, of course, be argued that the problem is not new, but merely heightened - for students have been employing algorithms whose workings they did not understand for centuries!).

How can students learn (be taught/encouraged) to look critically at the answers supplied? How much should students be required to know about the workings of black-boxes before being allowed to use them? For example, there are packages which invert matrices. If such a package uses floating-point arithmetic, it can give answers which should not be accepted at face value. At least students should be warned about this or, better, should learn to recognise when this has occured.

3.6 Cultural, social and economic factors

We have written of the computer as an aid to mathematics teaching and learning. So is the overhead projector. The difference though between the two tools is not, however, solely the enormous difference in the range of possibilities opened up by the former. Equally, it springs from the enormous effect which the computer is having upon society outside the confines of educational systems. As a result society has expectations concerning computers and their use, expectations which often have little basis in reality. Students too have expectations about their use. There are then enormous pressures on educators at all levels to use computers, not necessarily for their intrinsic value, but because society expects it, and not because to do so might be considered old-fashioned and reactionary.

It will be difficult for computers to be used effectively in education until society has become better informed about their power and limitations. Unrealistic expectations must be strongly discouraged. There is a danger that false advertising by computer companies and software developers, and a pressure from various sections of society, could lead to ill-designed, over-optimistic innovation and, in turn, to a backlash comparable with that of the 1970s resulting from the hasty introduction of the 'New Math'. Political decision makers in some countries are 'pushing' computers and computer-related curricula into education without adequate consideration of objectives and consequences.

It is important, therefore, to realise that:

- reasonable use of computers in education requires software programs and packages whose educational standards and qualities are comparable with the technical ones offered by the available hardware,
- integrating computers into the curriculum must be coordinated with teacher/faculty in-service, professional development programs,
- educational budgets must be prepared to permit appropriate expenditure on hardware, software, and teacher development,
- no curriculum should remain stagnant for a long period.

Not all problems associated with computers in education can be anticipated. Many questions need to be answered through research initiatives directed at investigating the possibilities, limitations and possible dangers of computer use in education. Some causes for concern are:

- uniformity in students' thinking and reasoning could arise from overuse of computers in the learning process,
- standardisation of software development (in an attempt to form a commercial market) may lead to mediocrity and conformity,
- subtleties of communication between teachers and students could be impoverished by overusing computers,
- insensitive working with computers could adversely influence the total intellectual development of students (of their intuitive thinking, creativity, perception, etc.).

The case of developing countries demands special attention. For them the provision and maintenance of hardware creates great problems. Moreover, scarce resources must be husbanded carefully. The computer could offer special advantages to them; on the other hand the absence or shortage of computers could widen still further the gap between them and the developed countries. Several conferences have considered the question of new technologies in education for developing countries (see, for example, Amara, Boudriga and Harzallah, 1986).

3.7 Conclusion

We are only experiencing the very beginning of the effect of computers on the teaching and learning of mathematics. Gradually, we are beginning to take advantage of some of their more obvious possibilities such as their quick and accurate production of graphical material, their quick and accurate (though not always precise) arithmetic, analyse large quantities of data.

In numerous publications one can see examples of mathematical situations for which the computer and informatics allow us to see and approach situations from a new point of view. Obvious examples which spring to mind are the many applications in statistics (dealing with vast quantities of data), in probability (with all the possibilities opened up for simulation by pseudo-random number generators); in geometry, too, there is a range of interesting activities – production and processing of images, curve plotting, the transformation of images (translations, reflections, etc.), loci, exploration of images and figures. The *dynamic* aspect dominates here: One can visualise instantly the effect of varying a parameter. In linear algebra, an algorithmic approach furnishes a tool both for calculations and also for demonstration. Here again the dynamic aspect plays an important role: To see a matrix steadily assume a diagonal form is very different from obtaining the result once and for all after a long and involved calculation. But it is above all in analysis that the opportunities to utilise informatics are richest and most numerous. The study of numbers, of functions, of the solution of equations, observation and study of sequences and series (and in particular of their speed of convergence), integral calculus, differential equations, asymptotic expansions, discretisation, power series for functions, etc. In addition to these 'classical' fields where the use of the computer arises naturally, one has also seen developments in newer fields which have occurred largely because of computers: Formal symbolic logic is a striking instance; discrete mathematics can provide us with other examples. The computer is not only an aid for computation and demonstration, but a force for development.

In all of these cases, the contribution of the computer takes several forms. Firstly, it is a calculating tool allowing numerous and rapid calculations; it also serves to place renewed emphasis on numerical methods, and thus on the study of algorithms; and, especially, it is a pedagogical tool for promoting teaching and learning.

However, let us reiterate, the act of using a computer does not automatically lead to an improvement. It is not a magic wand! Like all tools, it can serve us badly; we must learn how to get the best from it.

The development of technology (computers becoming smaller and cheaper) and the development of new tools (such as multimedia ones) will certainly provoke very large changes in education. Complex learning environments and integrated software will become more and more available. The technology age in education is still to come!

Computers are now widely to be found in schools and universities, but they are not always widely used. Teachers are being trained in their use, but principally in techniques and programming, and the question of giving them a true pedagogical training is not totally solved. It is also necessary to bear in mind that if we wish to change the educational system, then there will be a need simultaneously to reform both the training given to those preparing to teach in schools and universities and also the continuing education of existing teachers. Many interesting and rich experiments have been done, many enthusiastic teachers have produced activities and tools, and have tried new pedagogical strategies using computers. We now need to have ALL teachers able to use computers as a natural tool, and to integrate them into their teaching.

At the same time there is the need to carry out much research and experimentation so that we may effectively understand and control the impact of the use of the computer on students' learning and on their conceptions and representations of mathematical objects. Only after such studies will we be able to provide high quality software and, most importantly, a new range of didactical activities, tasks and situations to enhance learning.

REFERENCES

Banchoff, T. et al (Eds.) [1988]: *Educational Computing in Mathematics, ECM87*, Amsterdam: North Holland.

Amara, M., Boudriga, N. and Harzallah, K. (Eds.) [1986]: L'Informatique et l'enseignement des mathématiques dans les pays en voie de développement, *Proceedings of the first ICOMIDC Symposium*, Paris: ICOMIDC and UNESCO.

Cornu, B. [1992]: *L'Ordinateur pour enseigner les mathématiques*, Paris: Presses Universitaires de France.

Hirst, A. and Hirst, K. (Eds.) [1988]: *Proceedings of the Sixth International Congress on Mathematics Education, Budapest, 1988*, Budapest: Janos Bolyai Mathematical Society.

Johnson, D.C. and Lovis, F. (Eds.) [1987]: *Informatics and the Teaching of Mathematics*, Proceedings of the IFIP TC 3/WG 3.1 Working Conference on Informatics and the Teaching of Mathematics, Sofia, Bulgaria, 16 - 18 May, 1987. Amsterdam: North-Holland. ISBN 0-444-70325-X.

Tinsley, J.D. and van Weert, T.J. (Eds.) [1989]: *Educational Software at the Secondary Level*, Amsterdam: Elsevier.

7

LIVING WITH A NEW MATHEMATICAL SPECIES

Lynn Arthur Steen
St. Olaf College, Northfield, Minnesota 55077, U.S.A.

Computers are both the creature and the creator of mathematics. They are, in the apt phrase of Seymour Papert, "mathematics-speaking beings." J. David Bolter, in his stimulating book *Turing's Man* [Bolter, 1984], calls computers "embodied mathematics." Computers shape and enhance the power of mathematics, while mathematics shapes and enhances the power of computers. Each forces the other to grow and change, creating, in Thomas Kuhn's language, a new mathematical paradigm.

Until recently, mathematics was a strictly human endeavor. But suddenly, in a brief instant on the time scale of mathematics, a new species has entered the mathematical ecosystem. Computers speak mathematics, but in a dialect that is difficult for some humans to understand. Their number systems are finite rather than infinite; their addition is not commutative; and they don't really understand "zero," not to speak of "infinity." Nonetheless, they do embody mathematics.

The core of mathematics is changing under the ecological onslaught of mathematics-speaking computers. New specialties in computational complexity, theory of algorithms, graph theory, and formal logic attest to the impact that computing is having on mathematical research. As Arthur Jaffe has argued so well (in [Jaffe, 1984]), the computer revolution *is* a mathematical revolution.

New Mathematics for a New Age

Computers are discrete, finite machines. Unlike a Turing machine with an infinite tape, real machines have limits of both time and space. Theirs is not an idealistic Platonic mathematics, but a mathematics of limited resources. The goal is not just to get a result, but to get the best result for the least effort. Optimization, efficiency, speed, productivity—these are essential objectives of modern computer mathematics.

Computers are also logic machines. They embody the fundamental engine of mathematics—rigorous propositional calculus. The first celebrated computer proof was that of the four-color theorem: the computer served there as a sophisticated accountant, checking out thousands of cases of reductions. Despite philosophical alarms that computer-based proofs change mathematics from an *a priori* to a contingent, fallible subject (see, e.g., [Tymoczko, 1979]), careful analysis reveals that nothing much has really changed. The human practice of mathematics has always been fallible; now it has a partner in fallibility.

Research on the so-called Feigenbaum constant reveals just how far this evolution has progressed in just a few years: computer-assisted investigations of families of periodic maps suggested the presence of a mysterious universal limit, apparently independent of the particular family of maps. Subsequent theoretical investigations led to proofs that are true hybrids of classical analysis and computer programming [Eckmann, 1984], showing that computer-assisted proofs are possible not just in graph theory, but also in functional analysis.

Computers are also computing machines. By absorbing, transforming, and summarizing massive quantities of data, computers can simulate reality. No longer need the scientist build an elaborate wind tunnel or a scale model refinery in order to test engineering designs. Wherever basic science is well understood, computer models can emulate physical processes by carrying out instead the process implied by mathematical equations. Whereas mathematical models used to be primarily tools used by theoretical scientists to formulate general theories, now they are practical tools of enormous value in the everyday world of engineering and economics.

It has been just over fifty years since Alan Turing developed his seminal scheme of computability [Turing, 1936] in which he argued that machines could do whatever humans might hope to do. In abstract terms, what he proposed was a universal machine of mathematics (see [Hodges, 1983] for details). It took two decades of engineering effort to turn Turing's abstract ideas into productive real machines. During that same period abstract mathematics flourished, led by Bourbaki, symbolized by the "generalized abstract nonsense" of category theory. But with abstraction came power, with rigor came certainty. Once real computers emerged, the complexity of programs quickly overwhelmed the informal techniques of backyard programmers. Formal methods became *de rigueur*; even the once-maligned category theory is now enlisted to represent finite automata and recursive functions (see, e.g., [Beckman, 1984], [Lewis, 1981]). Once again, as happened before with physics, mathematics became more efficacious by becoming more abstract.

The Core of the Curriculum

Twenty-five years ago in the United States the Committee on the Undergraduate Program in Mathematics (CUPM) issued a series of reports that led to a gradual standardization of curricula among undergraduate mathematics departments [CUPM, 1965]. Shortly thereafter, in 1971, Garrett Birkhoff and J. Barkley Rosser presented papers at a meeting of the Mathematical Association of America concerning predictions for undergraduate mathematics in 1984. Birkhoff urged increased emphasis on modelling, numerical algebra, scientific computing, and discrete mathematics. He also advocated increased use of computer methods in pure mathematics: "Far from muddying the limpid waters of clear mathematical thinking, [computers] make them more transparent by filtering out most of the messy drudgery which would otherwise accompany the working out of specific illustrations." [Birkhoff, 1972, p. 651] Rosser emphasized many of the same points, and warned of impending disaster to undergraduate mathematics if their advice went unheeded: "Unless we revise [mathematics courses] so as to embody much use of computers, most of the clientele for these courses will instead be taking computer courses in 1984." [Rosser, 1972, p. 639]

In the first decade after these words were written, U.S. undergraduate and graduate degrees in mathematics declined by 50%. The clientele for traditional mathematics migrated to computer science, and the former CUPM consensus all but disappeared. In 1981 CUPM issued a new report, this one on the Undergraduate Program in Mathematical Sciences ([CUPM, 1981], reprinted in [CUPM, 1989]). Beyond calculus and linear algebra, they could agree on no specific content for the core of a mathematics major: "There is no longer a common body of pure mathematical information that every [mathematics major] should know."

The symbol of reformation became discrete mathematics. Anthony Ralston argued forcefully the need for change before both the mathematics community [Ralston, 1981] and the computer science community [Ralston, 1980]. Discrete mathematics, in Ralston's view, is the central link between the fields. The advocacy of discrete mathematics rapidly became quite vigorous (see, e.g., [Kemeny, 1983], [Ralston, 1983,] and [Steen, 1984]), and the Sloan Foundation funded experimental curricula at six institutions to encourage development of discrete-based alternatives to standard freshman calculus. The impact of this work can be seen in the growth of courses and publications: in the five year period from 1985 to 1990, hundreds of courses were created and over 40 new textbooks in discrete mathematics were published.

Soon calculus itself came under scrutiny, as a natural force for counter-reformation. Critics argued that the power of computation and the ubiquity of applications had changed fundamentally the role of calculus in the practice of mathematics (e.g., [Douglas, 1986; Steen, 1988]). The National Science Foundation launched diverse projects to reshape the nature of calculus instruction. Virtually all of these projects feature supporting roles for the numeric, symbolic, and graphic power of computers.

The need for consensus on the contents of undergraduate mathematics is perhaps the most important issue facing American college and university mathematics departments [CUPM, 1989]. On the one hand departments that have a strong traditional major often fail to provide their students with the robust background required to survive the evolutionary turmoil in the mathematical sciences. Like the Giant Panda, these departments depend for survival on a dwindling supply of bamboo—strong students interested in pure mathematics. On the other hand, departments offering flabby composite majors run a different risk: by avoiding advanced, abstract requirements, they often misrepresent the true source of mathematical knowledge and power. Like zoo-bred animals unable to forage in the wild, students who have never been required to master a deep theorem are ill-equipped to master the significant theoretical complications of real-world computing and mathematics.

Computer Literacy

Mathematical scientists at American institutions of higher education are responsible not only for the technical training of future scientists and engineers, but also for the technological literacy of the educated public—of future lawyers, politicians, doctors, educators, and clergy. Public demand that college graduates be prepared to live and work in a computer age has caused many institutions to introduce requirements in quantitative or computer literacy.

In 1981 the Alfred P. Sloan Foundation initiated curricular exploration of "the new liberal arts," the role of applied mathematical and computer sciences in the education of students outside technical fields. "The ability to cast one's thoughts in a form that makes possible mathematical manipulation and to perform that manipulation ...[has] become essential in higher education, and above all in liberal education." [Koerner, 1981, p. 6] Others echoed this call for reform of liberal education. David Saxon, President of the University of California wrote in a *Science* editorial that liberal education "will continue to be a failed idea as along as our students are shut out from, or only superficially acquainted with, knowledge of the kinds of questions science can answer and those it cannot." [Saxon, 1982]

Too often these days the general public views computer literacy as a modern substitute for mathematical knowledge. Unfortunately, this often leads students to superficial courses that emphasize vocabulary and experiences over concepts and principles [Steen, 1985]. The advocates of computer literacy conjure images of an electronic society dominated by the information industries. Their slogan of "literacy" echoes traditional educational values, conferring the aura but not the logic of legitimacy.

Typical courses in computer literacy are filled with ephemeral details whose intellectual life will barely survive the students' school years. These courses contain neither a Shakespeare nor a Newton, neither a Faulkner nor a Darwin; they convey no fundamental principles nor enduring truths. Computer literacy is more like driver education than like calculus. It teaches students the prevailing rules of the road concerning computers, but does not leave them well-prepared for a lifetime of work in the information age.

Algorithms and data structures are to computer science what functions and matrices are to mathematics. As much of the traditional mathematics curriculum is devoted to elementary functions and matrices, so beginning courses in computing—by whatever name—should stress standard algorithms and typical data structures. As early as students study linear equations they could also learn about stacks and queues; when they move on to conic sections and quadratic equations, they could in a parallel course investigate linked lists and binary trees.

Computer languages can (and should) be studied for the concepts they represent—procedures in Pascal and C, recursion and lists in Lisp—rather than for the syntactic details of semicolons and line numbers. They should not be undersold as mere technical devices for encoding problems for a dumb machine, nor oversold as exemplars of a new form of literacy. Computer languages are not modern equivalents of Latin or French; they do not deal in nuance and emotion, nor are they capable of persuasion, conviction, or humor. Although computer languages do represent a new and powerful way to think about problems, they are not a new form of literacy.

Computer Science

In the United States, computer science programs cover a broad and varied spectrum, from business-oriented data processing curricula, through management information science, to theoretical computer science. All of these intersect with the mathematics curriculum, each in different ways.

To help clarify these conflicting approaches, Mary Shaw of Carnegie Mellon University put together a composite report on the undergraduate computer science curriculum. This report is quite forceful about the contribution mathematics makes to the study of computer science: "The most important contribution a mathematics curriculum can make to computer science is the one least likely to be encapsulated as an individual course: a deep appreciation of the modes of thought that characterize mathematics." [Shaw, 1984, p. 55]

The converse is equally true: one of the more important contributions that computer science can make to the study of mathematics is to develop in students an appreciation for the power of abstract methods when applied to concrete situations. Students of traditional mathematics used to study a subject called "Real and Abstract Analysis;" students of computer science now can take a course titled "Real and Abstract Machines." In the former "new math," as well as in modern algebra, students learned about relations, abstract versions of functions; today business students study "relational data structures" in their computer courses, and advertisers tout "fully relational" as the latest innovation in business software.

An interesting and pedagogically attractive example of the power of abstraction made concrete can be seen in the popular electronic spreadsheets that are marketed under such trade names as Lotus and Excel. Originally designed for accounting, they can as well emulate cellular automata or the Ising model for ferromagnetic materials [Hayes, 1983]. They can also be "programmed" to carry out most standard mathematical algorithms—the Euclidean algorithm, the simplex method, Euler's method for solving differential equations [Arganbright, 1985]. An electronic spreadsheet—the archetype of applied computing—is a structured form for recursive procedures—the fundamental tool of algorithmic mathematics. It is a realization of abstract mathematics, and carries with it much of the power and versatility of mathematics.

Computers in the Classroom

Just as the introduction of calculators upset the comfortable pattern of primary school arithmetic, so the spread of computers will upset the traditions of secondary and tertiary mathematics. This year long division is passe; next year integration will be under attack.

The impact of computing on secondary school mathematics has been the subject of many discussions in the United States (e.g., [Steen, 1987]). Jim Fey, coordinator of two assessments ([Corbitt, 1985; Fey, 1984]), described these efforts as "an unequivocal dissent from the spirit and substance of efforts to improve school mathematics that seek broad agreement on conservative curricula." [Fey, 1984, p. viii] The new *Curriculum and Evaluation Standards for School Mathematics* [NCTM, 1989] of the National Council of Teachers of Mathematics as well as other recommendations from the U.S. National Academy of Sciences ([NRC, 1989; MSEB, 1990]) set expectations for school mathematics that employ calculators and computers in every appropriate manner.

Teachers in tune with the computer age seek change in both curriculum and pedagogy. But the inertia of the system remains high. For example, the 1982 International Assessment of Mathematics documented that in the United States calculators are never permitted in one-third of the 8th grade classes, and rarely used in all but 5% of the rest [McKnight, 1987]. Recent data [NAEP, 1991] show some improvement, but still fall well short of the NCTM recommendations.

Laptop computers are now common—they cost about as much as ten textbooks, but take up only the space of one. Herb Wilf argues (in [Wilf, 1982]) that it is only a matter of time before students will carry with them a device to perform all the algorithms of undergraduate mathematics. Richard Rand, in a survey [Rand, 1984] of applied research based on symbolic algebra agrees: "It will not be long before computer algebra is as common to engineering students as the now obsolete slide rule once was." Just five years after Wilf's article appeared, the same journal carried a review [Nievergelt, 1987] of the first pocket calculator with symbolic algebra capabilities.

Widespread use of computers that do school and college mathematics will challenge standard educational practice [Steen, 1990]. For the most part, computers reinforce the student's desire for correct answers. In the past, their school uses have primarily extended the older "teaching machines:" programmed drill with pre-determined branches for all possible responses. But the recent linking of symbolic algebra programs with so-called "expert systems" into sophisticated "intelligent tutors" has produced a rich new territory for imaginative computer-assisted pedagogy that advocates claim can rescue mathematics teaching from the morass of rules and template-driven tests (see e.g., [Smith, 1988; Zorn, 1987]).

It is commonplace now to debate the wisdom of teaching skills (such as differentiation) that computers can do as well or better than humans. Is it really worth spending one month of every year teaching half of a country's 18-year-old students how to imitate a computer? What is not yet so common is to examine critically the effect of applying to mathematics pedagogy computer systems that are themselves only capable of following rules or matching templates. Is it wise to devise sophisticated computer systems to teach efficiently precisely those skills that computers can do better than humans, particularly those skills that make the computer tutor possible? In other words, since computers can now do the calculations of algebra and calculus, should we use this power to reduce the curricular emphasis on calculations or to make the teaching of these calculations more efficient? This is a new question, with a very old answer.

Let Us Teach Guessing

Forty years ago George Pólya wrote a brief paper with the memorable title "Let Us Teach Guessing" [Pólya, 1950]. It is not differentiation that our students need to learn, but the art of guessing. A month spent learning the rules of differentiation reinforces a student's ability to learn (and live by) the rules. In contrast, time spent making conjectures about derivatives will teach a student something about the art of mathematics and the science of order.

With the aid of the mathematics-speaking computer, students can for the first time learn college mathematics by discovery. This is an opportunity for pedagogy that mathematics educators cannot afford to pass up. Mathematics is, after all, the science of order and pattern, not just a mechanism for grinding out formulas. Students discovering mathematics gain insight into the discovery of pattern, and slowly build confidence in their own ability to understand mathematics. Formerly, only students of sufficient genius to forge ahead on their own could have the experience of discovery. Now with computers as an aid, the majority of students can experience for themselves the joy of discovery.

Metaphors for Mathematics

Two metaphors from science are useful for understanding the relation between computer science, mathematics, and education. Cosmologists long debated two theories for the origin of the universe—the Big Bang theory, and the theory of Continuous

Creation. Current evidence tilts the cosmology debate in favor of the Big Bang. Unfortunately, this is all too often the public image of mathematics as well, even though in mathematics the evidence favors Continuous Creation.

The impact of computer science on mathematics and of mathematics on computer science is the most powerful evidence available to beginning students that mathematics is not just the product of an original Euclidean big bang, but is continually created in response to challenges both internal and external. Students today, even beginning students, can learn things that were simply not known twenty years ago. We must not only teach new mathematics and new computer science, but we must teach as well the fact that this mathematics and computer science is new. That's a very important lesson for the public to learn.

The other apt metaphor for mathematics comes from the history of the theory of evolution. Prior to Darwin, the educated public believed that forms of life were static, just as the educated public of today assumes that the forms of mathematics are static, laid down by Euclid, Newton, and Einstein. Students learning mathematics from contemporary textbooks are like the pupils of Linnaeus, the great eighteenth-century Swedish botanist: they see a static, pre-Darwinian discipline that is neither growing nor evolving. Learning mathematics for most students is an exercise in classification and memorization, in labeling notations, definitions, theorems, and techniques that are laid out in textbooks as so much flora in a wondrous if somewhat abstract Platonic universe.

Students rarely realize that mathematics continually evolves in response to both internal and external pressures. Notations change; conjectures emerge; theorems are proved; counterexamples are discovered. Indeed, the passion for intellectual order combined with the pressure of new problems—especially those posed by the computer—force researchers to continually create new mathematics and archive old theories.

The practice of computing and the theory of computer science combine to change mathematics in ways that are highly visible and attractive to students. This continual change reveals to students the living character of mathematics, restoring to the educated public some of what the experts have always known—that mathematics is a living, evolving component of human culture.

REFERENCES

Arganbright, D.E. [1985]: *Mathematical Applications of Electronic Spreadsheets*, New York: McGraw-Hill.

Beckman, F.S. [1984]: *Mathematical Foundations of Programming*. The Systems Programming Series, Reading, MA: Addison-Wesley.

Birkhoff, G. [1972]: The Impact of Computers on Undergraduate Mathematics Education in 1984, *Amer. Math. Monthly*, 79, 648-657.

Bolter, J.D. [1984]: *Turing's Man: Western Culture in the Computer Age*, Chapel Hill, NC: University of North Carolina Press.

Committee on the Undergraduate Program in Mathematics [1965]: *A General Curriculum in Mathematics for Colleges*, Washington, DC: Mathematical Association of America.

Committee on the Undergraduate Program in Mathematics [1981]: *Recommendations for a General Mathematical Sciences Program*, Washington, DC: Mathematical Association of America.

Committee on the Undergraduate Program in Mathematics [1989]: *Reshaping College Mathematics*. MAA Notes No. 13, Washington, DC: Mathematical Association of America.

Corbitt, M.K., and Fey, J.T. (Eds.) [1985]: *The Impact of Computing Technology on School Mathematics: Report of an NCTM Conference*, Reston, VA: National Council of Teachers of Mathematics.

Douglas, R.G. (Ed.) [1986]: *Toward A Lean and Lively Calculus*. MAA Notes Number 6, Washington, DC: Mathematical Association of America.

Eckmann, J.-P., Koch, H. and Wittwer, P. [1984]: A Computer-assisted Proof of Universality for Area-preserving Maps, *Memoirs of the American Mathematical Society*, 47:289 (January).

Fey, J.T., *et al.* (Eds.) [1984]: *Computing and Mathematics: The Impact on Secondary School Curricula*, Reston, VA: National Council of Teachers of Mathematics.

Hayes, B. [1983]: Computer Recreations, *Scientific American*, 22-36, (October).

Hodges, A. [1983]: *Alan Turing: The Enigma*, New York: Simon and Schuster.

Jaffe, A. [1984]: Ordering the Universe: The Role of Mathematics in *Renewing U.S. Mathematics*, Washington, DC: National Academy Press.

Kemeny, J.G. [1983]: Finite Mathematics—Then and Now in Ralston, A., and Young, G.S. (Eds.), *The Future of College Mathematics*, pp. 201-208, New York: Springer-Verlag.

Koerner, J.D. (Ed.) [1981]: *The New Liberal Arts: An Exchange of Views*, New York: Alfred P. Sloan Foundation.

Lewis, H.R., and Papadimitriou, C.H. [1981]: *Elements of the Theory of Computation*, Englewood Cliffs, NJ: Prentice-Hall.

Mathematical Sciences Education Board [1990]: *Reshaping School Mathematics: A Philosophy and Framework for Curriculum*, Washington, DC: National Academy Press.

McKnight, C.C., *et al* [1987]: *The Underachieving Curriculum: Assessing U.S. School Mathematics from an International Perspective*, Champaign, IL: Stipes.

National Assessment of Education Progress [1991]: *The State of Mathematics Achievement*, Washington, DC: U.S. Department of Education.

National Council of Teachers of Mathematics [1989]: *Curriculum and Evaluation Standards for School Mathematics*, Reston, VA: National Council of Teachers of Mathematics.

National Research Council [1989]: *Everybody Counts: A Report to the Nation on the Future of Mathematics Education*, Washington, DC: National Academy Press.

Nievergelt, Y. [1987]: The Chip with the College Education: The HP-28C, *Amer. Math. Monthly*, 94, 895-902.

Pólya, G. [1950]: Let Us Teach Guessing, *Etudes de Philosophie des Sciences*, Neuchatel: Griffon, pp. 147-154; reprinted in G. Pólya, *Collected Papers*, Vol. IV, Cambridge, MA: MIT Press, 1984, pp. 504-511.

Ralston, A. [1981]: Computer Science, Mathematics, and the Undergraduate Curricula in Both, *Amer. Math. Monthly*, 88, 472-485.

Ralston, A., and Shaw, M. [1980]: Curriculum '78: Is Computer Science Really that Unmathematical?, *Communications of the ACM*, 23, 67-70.

Ralston, A., and Young, G.S. (Eds.) [1983]: *The Future of College Mathematics*, New York: Springer-Verlag.

Rand, R.H. [1984]: *Computer Algebra in Applied Mathematics: An Introduction to MACSYMA*, London: Pitman.

Rosser, J.B. [1972]: Mathematics Courses in 1984, *Amer. Math. Monthly*, 79, 635-648.

Saxon, D.S. [1982]: Liberal Education in a Technological Age, *Science*, 218, 845 (26 November).

Shaw, M. (Ed.) [1984]: *The Carnegie-Mellon Curriculum for Undergraduate Computer Science*, New York: Springer-Verlag.

Smith, D.A.; Porter, G.J.; Leinbach, L.C.; and Wenger, Ronald H. (Eds.) [1988]: *Computers and Mathematics: The Use of Computers in Undergraduate Instruction*. MAA Notes No. 9, Washington, DC: Mathematical Association of America.

Steen, L.A. [1984]: 1 + 1 = 0: New Math for a New Age, *Science*, 225, 981 (7 September).

Steen, L.A. [1991]: Twenty Questions for Computer Reformers in Demana, F., et. al. *Proceedings of the Conference on Technology in Collegiate Mathematics*, Reading, MA: Addison-Wesley, pp. 16-19.

Steen, L.A. (Ed.) [1988]: *Calculus for a New Century: A Pump, Not a Filter*, Washington, DC: Mathematical Association of America.

Steen, L.A. [1987]: Who Still Does Math with Paper and Pencil?, *Chronicle of Higher Education*, A48 (14 October).

Steen, L.A. [1985]: Mathematics and Computing Education: A Common Cause, *Communications of the ACM*, 28, 666-667.

Turing, A.M. [1936]: On Computable Numbers, with an Application to the Entscheidungsproblem, *Proc. London Math. Soc.*, 2nd Ser., 42, 230-265.

Tymoczko, T. [1979]: The Four Color Problem and its Philosophical Significance, *Journal of Philosophy*, 76, 57-85.

Wilf, H.S. [1982]: The Disk with the College Education, *Amer. Math. Monthly*, 89, 4-8.

Zorn, P. [1987]: Computing in Undergraduate Mathematics, *Notices of the American Mathematical Society*, 34, 917-923 (October).

WHAT ARE ALGORITHMS? WHAT IS ALGORITHMICS?

Stephen B. Maurer
Swarthmore College, Swarthmore, Pennsylvania 19081-1397, U.S.A.

Overview. Roughly speaking, an algorithm is a precise, systematic method for solving some class of problems. Algorithmics is the systematic study of algorithms – how to devise them, describe them, validate them and compare their relative merits. There have been algorithms in mathematics since ancient times, but algorithmics is new. Only with the advent of computers has it been possible to tackle such large and complicated problems that a systematic approach to algorithms is necessary. Because algorithms are now essential in almost all business and scientific applications of mathematics (as well as being increasingly important to mathematicians themselves and fundamentally important to computer scientists), it is important that mathematics education take algorithms and algorithmics into account.

This paper has four sections. In Section 1, by far the longest, we explain what algorithms are in much more detail, presenting many examples. In Section 2 we do the same for algorithmics. In Section 3 we discuss several reasons why the study of algorithms and algorithmics is valuable in mathematics, and we also discuss some counterarguments. Finally, in Section 4 we make some suggestions for incorporating algorithms and algorithmics into the secondary and tertiary mathematics curriculum.

1. What Are Algorithms?

Algorithms turn *input* data into *output* data through sequences of actions. For instance, an algorithm might take two integers and output their product. The rules specifying the algorithm (including rules specifying what inputs are allowed) must be precise enough to satisfy

1. Determinateness. For each allowed input, the first action is uniquely determined, and more generally, after each action in the sequence the successor action is uniquely determined.

It doesn't do us any good to have an algorithm that doesn't stop, so we also require

2. Finiteness. For any allowed input, the algorithm must stop after a finite sequence of actions.

Usually algorithms are devised to solve problems. Such algorithms must be appropriate for the purpose at hand:

3. Conclusiveness. When the algorithm terminates, it must either output a solution to the problem for the given input, or it must indicate that it cannot solve the problem.

In some cases it is reasonable to relax these stringent requirements; we'll take up this point later. One can also ask: how precise is precise? Just how are the rules to be stated to make them precise? Good question. It depends on who or what you are talking to. We will also address this further. But let's turn immediately to some examples.

Example 1: Arabic Multiplication

The traditional paper and pencil algorithm for multiplying two numbers expressed in arabic numerals is brilliant. Too bad we all take it for granted. It's brilliant because it reduces a general problem to a small subcase – how to multiply two single-digit integers – and does so in a small amount of space. Here's the result of applying the algorithm to 432 × 378:

$$\begin{array}{r} 432 \\ 378 \\ \hline 3456 \\ 3024 \\ 1296 \\ \hline 163296 \end{array}$$

Each row of intermediate calculation is obtained by multiplying the top factor (432) by one digit of the bottom factor. If we expand out the first intermediate row in more detail, we get

$$\begin{array}{r} 432 \\ 8 \\ \hline 16 \\ 24 \\ 32 \\ \hline 3456 \end{array} \tag{1}$$

Of course, it's never written this way. To save space, the "carries" are either all done mentally, or they are marked with small digits as follows:

$$3\quad 4\quad \overset{2}{5}\quad \overset{1}{6}$$

We include Display (1) to make the role of single-digit multiplications explicit. For instance, 16 is the product of the 2 in 432 by the 8 in 378.

Now, is this format precise enough for presenting Arabic multiplication? Apparently so, because such a format does seem to suffice for teaching the algorithm to children (when presented with many examples, lots of oral explanations, and hands-on practice). And you do need to make use of diagrams if the physical positioning of symbols on the page is part of the algorithm.

Nonetheless, this is not the format we will use for other algorithms, and it is not a good format for systematically verifying the defining conditions 1–3 above. So we now restate Arabic multiplication using "algorithmic language", a language style quite similar to a programming language. (We will assume basic familiarity with how such languages are to be read, e.g., what a loop is, what an assignment is.) Actually, we restate only the part shown in Display (1) – a multidigit number times a single digit number. The algorithm makes use of two procedures,

DigMult(a, b)	which multiplies the single digits a and b and returns M_l and M_r, the left and right digits of the product.
DigAdd(a, b)	which adds the single digits a and b and returns A_l and A_r, the left and right digits of the sum. (A_l will be either 0 or 1.)

Here's the algorithm:

```
Input $a_0, a_1, \ldots, a_m$   [the ones, tens,..., digits
                                 of an $(m+1)$-digit number]
      $b$                       [the one-digit multiplier]
Algorithm
  carry $\leftarrow 0$
  for $j = 0$ to $m$
    DigMult($a_j, b$)
    DigAdd($M_r$, carry)   [add any carry from
                            previous product]
    $P_j \leftarrow A_r$   [$j$th digit of the product known]
    DigAdd($M_l, A_l$)  [needed in case the carry
                         affects the left digit]
    carry $\leftarrow A_r$   [carry to the next single-
                              digit multiplication]
  endfor
  if carry $> 0$ then $P_{m+1} \leftarrow$ carry
Output $P_0, P_1, \ldots, P_m$ and sometimes $P_{m+1}$
                                 [digits of the product]
```

This is no doubt hard to follow, but try carrying it out on the example above. Look at 432 × 7 (the middle line of the first example), which shows why the two lines before "endfor" are needed. That this description is hard to follow should bring home the point that the Arabic algorithm is really quite subtle. (For instance, we don't include a step just before endfor to carry A_l, because at this point A_l is always 0. Do you see why?)

The advantage of this formulation of the algorithm is that it is easier to verify that it *is* an algorithm. Is it determinate? Yes, because each line leaves no doubt about what is to be done, and the order of execution is also specified – go down the page, except when you get to the end of a loop, go back to the beginning. Is it finite? Yes, because the loop has only 5 lines, and the loop gets carried out $m + 1$ times. Does it solve the problem? This is not so obvious, but the specificity of the lines makes it easier to present a proof when it is time to get around to that. (We will talk about algorithm verification later.)

Notice that this algorithm involves *iteration*: some subprocess is applied repetitively. In this case the subprocess of multiplying two single-digit numbers (and then carrying) was iterated. While an algorithm does not have to involve iteration (or a related type of repetition called recursion), almost all algorithms of interest in mathematics do.

Example 2: Euclid's Algorithm

This one is much older than the first, and also much simpler, but perhaps not so well known. It is the classical Greek method for finding the greatest common divisor (gcd) of two positive integers. It assumes you already know how to divide and find remainders. The algorithm keeps dividing and finding a remainder until the remainder is 0. Then the *previous* remainder is the gcd of the original numbers.

Here is a numerical example. Find the gcd of 147 and 33. The quotient of 147 divided by 33 is 4 with remainder 15. That is,

$$147 = 33 \times 4 + 15.$$

So any number that divides 147 and 33 also divides 15, and conversely, any number that divides 33 and 15 divides 147. Now, do the same operations to 33 and 15 that we did to 147 and 33: 15 divides into 33 with remainder 3. Thus a number divides 33 and 15 if and only if it divides 15 and 3. But 3 divides into 15 exactly. So the largest number dividing 3 and 15 is 3 itself. Thus the gcd of 147 and 33 is 3.

In algorithmic language, Euclid's algorithm is the following:

```
Input $m, n$                                  [integers >= 0]
Algorithm
    num ← $m$; denom ← $n$
    repeat until denom = 0
        quot ← ⌊num/denom⌋
                                   [integer part of num/denom]
        rem ← num − quot*denom
        num ← denom; denom ← rem
                                     [update num and denom]
    endrepeat
Output num
```

For instance, for the numerical example above, initially num(erator) is 147 and denom(inator) is 33. Since $33 \neq 0$, we enter the repeat loop, quot(ient) is computed as 4 and rem(ainder) as 15. Then (33,15) become the new (num,denom) pair. Since denom is still not 0, we traverse the loop again, and (num,denom) becomes (15,3). At this point, working by hand, we immediately recognized that 3 divides 15, but a computer must "discover" this by following the rules. Since $3 \neq 0$, we enter the loop again, and update (num,denom) to (3,0). Now denom = 0 and the algorithm quits, outputting num = 3 as the gcd.

Notice there is no factoring in this algorithm. Another way to find $\gcd(m, n)$ is to factor m and n, and then take the product of all common factors. This second method is the standard one currently taught in elementary schools in North America. For small values of m and n, the second method is often faster than Euclid's method, but factoring very large numbers is very hard. In general, Euclid's method is the way to go.

Euclid's method is an algorithm. Clearly it is determinate. It is finite, because rem is always a nonnegative integer and gets smaller with each iteration, so eventually it must reach 0 and the algorithm stops. The algorithm is conclusive (correctly determines the gcd) for the reasons we argued informally above. A formal proof would be by mathematical induction.

Example 3: Matrix Multiplication

Let A be an $m \times n$ matrix and B an $n \times p$ matrix. Call the entry of A in row i (down from the top) and column j (from the left) a_{ij}. Similarly, $B = [b_{jk}]$. Then their product AB is defined to be the $m \times p$ matrix whose (i, k) entry is

$$\sum_{j=1}^{n} a_{ij} b_{jk}. \tag{2}$$

For instance,

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & -6 \\ -5 & 4 \\ 3 & -2 \end{bmatrix} = \begin{bmatrix} 6 & -4 \\ 21 & -16 \end{bmatrix}.$$

In particular, the (2,1) entry of the product is

$$4 * 7 + 5 * (-5) + 6 * 3 = 21.$$

How can we express the definition of matrix multiplication as an algorithm?

Informally, you just go through each combination of a row from A and a column from B and compute their product according to (2). Their product is a sum of real-number products, so we can compute it by keeping a running sum and successively adding real products until we are done. In algorithmic language we have

```
Input $A, B, m, n, p$
Algorithm
    for $i = 1$ to $m$
        for $k = 1$ to $p$
            $c_{ik} \leftarrow 0$
                        [initialize the $ik$ entry of $C = AB$]
            for $j = 1$ to $n$
                $c_{ik} \leftarrow c_{ik} + a_{ij} * b_{jk}$
            endfor
        endfor
    endfor
Output $C$
```

Example 4: Construct $\sqrt{n}$

All the examples so far have been arithmetic or algebraic. Here's one from geometry. By constructing a number r, we shall mean constructing a line segment of length r, starting with a line segment of length 1 and using a straightedge and compass. To construct $\sqrt{2}$, construct a unit perpendicular at one end of the initial unit segment. By the Pythagorean Theorem, the hypotenuse has length $\sqrt{2}$. Now it is possible to construct $\sqrt{3}$ by repeating the process. Construct a unit perpendicular at the end of the segment of length $\sqrt{2}$. The new hypotenuse will have length $\sqrt{3}$. See Figure 1.

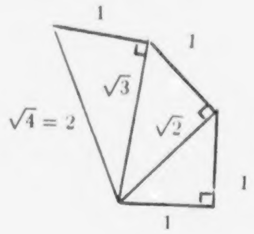


Figure 1

By induction, it should be clear that $\sqrt{n}$ may be constructed for all positive integers n. Here is the construction in algorithmic language.

```
Input n, unit line segment AB
Algorithm
  for c = 2 to n
    Construct BC ⊥ AB, with BC = 1
    AB ← AC                          [change names]
  endfor
Output AB                  [segment of length √n]
```

Is this determinate enough to be called an algorithm? It depends on the audience. If the reader knows well how to construct perpendiculars with straightedge and compass, it is. If not, the line "Construct $BC \perp AB$" must be expanded.

Example 5: Towers of Hanoi

Algorithmic approaches apply not just to traditional mathematical topics, but also to any situation where a systematic and repetitive approach is needed for a solution. Towers of Hanoi (TOH) is a game played with a set of n rings (or disks) of different sizes and three poles. Initially the rings are all on one pole, from smallest on top to largest on the bottom. The object is to get them all to another pole, in the same order, making moves according to the following rules.

1. Move only one ring at a time.
2. A larger ring may never be placed on a smaller ring.

TOH is often used by psychologists doing experiments with children. While it is easy to figure out solutions for $n = 3$ or 4, for larger n most kids soon lose their way. University students often don't do much better! The key to understanding why the game can be solved is *recursion* – reduce to the previous case. Suppose we already know how to solve the $(n-1)$-ring game. Regarding that subgame as an indivisible block, then Figure 2 shows how to solve the n-ring game. This solution may be put into algorithmic language if we allow a procedure (recall DigMult in Example 1) to invoke itself. The procedure H in the algorithm is first defined (in terms of itself) and then invoked by the (one-line) main algorithm. The poles are numbered 1,2,3. Note, therefore, that if r and s are numbers of two different poles, then $6 - r - s$ is the number of the third pole.

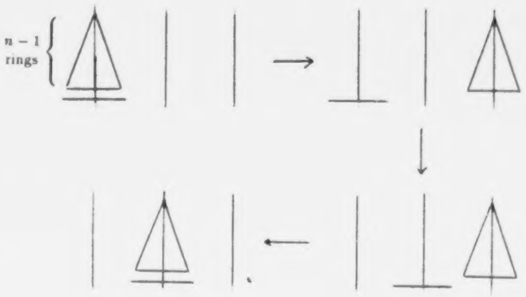


Figure 2

```
Input num, Pinit, Pfin          [number of disks,
          initial pole number, final pole number]
Algorithm
  procedure H(n, r, s)
                   [move n disks from pole r to pole s]
    if n = 1 then Move disk on r to s
             else H(n−1, r, 6−r−s)
     [move all but bottom disk to nontarget pole]
                  Move disk on r to s
                  H(n−1, 6−r−s, s)
              [move other disks onto target pole]
    endif
  endprocedure
  H(num, Pinit, Pfin)
                      [main algorithm – invoke H]
Output Solution to the game
```

That this is an algorithm is not so clear. It's not clear how to start carrying out the call of H, since mostly it just calls itself again instead of moving disks. It's also not clear that when it finishes (if it finishes), it has solved the game. But in fact it is an algorithm, and once one develops a good understanding of how recursion works, it is fairly evident why. In any event, good programming languages have recursion built in, and thus the algorithm above is easy to translate into such languages.

Example 6: The Quadratic formula

The traditional formula for solving $ax^2+bx+c = 0$ seems simple enough; where's the algorithm and why bother with it? Well, there are several cases – two distinct real roots, one repeated real root, no real roots – and properly choosing between cases is an algorithmic matter. Even if the audience knows about complex numbers, if they want to compute solutions, there is the problem that most calculators and computers won't accept a request to take the square root of a negative number. So presenting the solution process as an algorithm has merit.

```
Input $a, b, c$                        [coefficients of
                              $ax^2 + bx + c$, with $a \neq 0$]
Algorithm
  $D \leftarrow b^2 - 4ac$
  if                                  [three cases follow]
    $D > 0$ then                         [two real roots]
           $s \leftarrow \sqrt{D}$
           $x_1 \leftarrow (b+s)/2a$
           $x_2 \leftarrow (b-s)/2a$
    $D = 0$ then $x_1 \leftarrow x_2 \leftarrow b/2a$
                                [one repeated real root]
    $D < 0$ then                      [two complex roots]
           $s \leftarrow \sqrt{-D}$
           $x_1 \leftarrow (b+is)/2a$
           $x_2 \leftarrow (b-is)/2a$
  endif
Output the roots, $x_1$ and $x_2$
```

If we want to be even more comprehensive, and allow input with $a = 0$, then we have to include several more cases. Note that there are no loops in this algorithm, but several if-statements (even more if $a = 0$ is allowed). Many procedures in the everyday world involve more multiple decisions than iteration – think of tax laws. Such procedures translate into algorithms with many if-statements.

Example 7: Numerical Solution of Equations

There is no formula for most equations $f(x) = 0$ that need to be solved in real applications, so one must use numerical approximations. A common approach is the ***bisection method***. If $f(x)$ is continuous, and one can find input values a and b with $f(a) < 0$ and $f(b) > 0$, then there is at least one root in between. ($f(a) > 0$ and $f(b) < 0$ is just as good, and below we cover both cases by the condition $f(a)f(b) < 0$.) Try the midpoint $c = (a + b)/2$. It is unlikely that $f(c) = 0$, but the sign of $f(c)$ tells us which half of the interval $[a, b]$ to look in further. Now iterate:

```
Input $a, b$                              [$f(a)f(b) < 0$]
Algorithm
  repeat
    $c \leftarrow (a + b)/2$
    if $f(c) = 0$ then exit
    if $\text{sign}(f(c)) = \text{sign}(f(a))$
        then $b \leftarrow c$
        else $a \leftarrow c$
  endrepeat
Output $c$
```

Now, this is *not* an algorithm, because it can go on forever. For instance, if $f(x) = x^2 - 2$, $a = 1$ and $b = 2$, then it takes an infinite number of halvings to converge to the root $c = \sqrt{2}$. Of course, a cutoff condition can be added:

endrepeat when $|a-b| <$ tolerance

for whatever tolerance you choose. Even with such a condition, a real computer running this algorithm may not terminate, because, if the tolerance chosen is very small, roundoff error may result in $|a-b| >$ tolerance no matter how many iterations are performed.

Nonetheless, it may be best to present this algorithm initially in the nonterminating form above – it gets at the key idea of bisection without obscuring details, and it also ties in with the concepts of infinite processes and limits needed for a full mathematical attack. So this is our first example that suggests why the three defining conditions at the start of this section should often be relaxed.

Example 8: Sequences of Heads and Tails

An important role of mathematics is to guide us in making decisions under uncertainty. This can often be done using probability theory, but often the most direct approach is simulation. To take a very simple example, suppose we flip a fair coin until we get two heads in a row. How many flips should we expect to take? If we actually carry out this experiment many times, we find out what to expect. Here is a algorithm to carry out the experiment one time. Rand(0,1) is a command for flipping a coin; the output 1 means heads, 0 means tails. The algorithm could be run a thousand times inside a loop of a bigger algorithm, which could then analyze the output data in various ways (take the average, the variance, draw graphs, etc).

```
Input (none)
Algorithm
  count ← headct ← 0
  repeat
    flip ← Rand(0,1)          [0 or 1, at random]
    count ← count + 1
    if flip = 1
       then headct ← headct + 1
       else headct ← 0
  endrepeat when headct = 2
Output count
                 [total flips to get 2 heads in a row]
```

Now, this algorithm violates our definition in two ways. First, it is not determinate: actions are not uniquely determined. Second, it is theoretically possible that it won't terminate – we might get 0s forever. Nonetheless, we certainly want to be able to study such "algorithms". The hard part, actually, is to get computers to perform such procedures, since computers really are determinate machines. In other words, how can computers be made to produce what appear to be random numbers? Fortunately, there are good answers, using "pseudo-random number generators".

Example 9: A Calculator Exercise

Except in Example 1 we have not said anything about how our calculations are carried out; it could be by hand, by calculator, or by computer. In fact, what is easy to do depends on the device. In this example let us specifically consider hand calculators, since one can hope that this product of modern technology can be made available to students almost worldwide.

Consider the problem:

$$\text{evaluate } a(b_1 + b_2 + \cdots + b_n).$$

How shall we do this? On my scientific calculator, which has parentheses buttons, I can do it exactly in the order presented.

$$a \times (b_1 + b_2 + \cdots + b_n) = \tag{3}$$

where each symbol now represents a button (except a, b_1, etc., may represent many number buttons, and $\ldots$ represents repetition). However, we can save time if we multiply by a on the right:

$$b_1 + b_2 + \cdots + b_n = \times a = \tag{4}$$

A "+" would do as well as the first "="; the point is, the sum is computed as we go along, so once the sum is finished, we can proceed to multiplication. One button-push is saved. Also, if you have only a simple 4-function calculator without parentheses buttons, approach (4) is available while approach (3) is not.

Still other approaches are possible. Using the distributive law, we could instead evaluate

$$ab_1 + ab_2 + \cdots + ab_n.$$

The direct approach to this, using the fact that multiplications are completed before addition (on my calculator), is

$$a \times b_1 + a \times b_2 + \cdots + a \times b_n =$$

which involves considerably more button-pushes. But many calculators, mine included, have a feature to shorten repeated multiplication by the same factor: hit the × button twice. Thus the following string of steps displays first ab_1, then ab_2, and so on:

$$a \times \times b_1 = b_2 = b_3 = \cdots$$

Now we want to add these up, but hitting + (or any other operator on the main display) will cancel the effect of × ×. So instead we push M+, the memory plus button, which does the addition in the hidden memory register. Finally, at the end, we push MR to remove memory:

$$a \times \times b_1 \ \text{M+} \ b_2 \ \text{M+} \cdots b_n \ \text{M+} \ \text{MR} \tag{5}$$

Perhaps this sequence looks sufficiently odd that a presentation in algorithmic language would help:

```
push $a$ × × $b_1$ M+
for $k = 2$ to $n$
   push $b_k$ M+
endfor
push MR
```

A count shows that method (5) takes the same number of button-pushes as the original approach (3), and only one more than the best approach (4). So this problem provides a good example of how the issue of relative efficiency of algorithms pertains to even very elementary mathematics.

To close this section, let us emphasize that by algorithms we do not mean computer programs. We mean procedures for solving problems presented in a sufficiently precise form for careful analysis. While we have written most of our algorithms in a style which until recently has been associated only with computer programs, this is because that style is a good one for making key points precise. Our algorithm descriptions cannot be input directly to any computer. They omit all sorts of information that a computer would need to know about (how is the data input and output, what type of variables need to be declared, how much storage must be reserved?). Many computer scientists call this sort of algorithm description ***pseudocode***, because it is not real code for computers. But it is quite real for the sort of communication that interests us here – between humans – and so we prefer to call it ***algorithmic language***.

2. What Is Algorithmics?

First, algorithmics does ***not*** mean performing a lot of algorithms. Students worldwide have suffered too much rote repetition of mathematical algorithms over the years already. In the future, algorithms will be carried out more and more by machines, or by person-machine combinations, so hand calculation except of the simplest sort should receive less emphasis.

Algorithmics *is* the process of creating, understanding, validating and comparing algorithms. In short, it is thinking ***about*** algorithms, not thinking ***like*** algorithms.

Here is another way to put this. The phrase "algorithmic mathematics" has two meanings, traditional and contemporary [Maurer, 1984]. The traditional meaning emphasizes carrying out algorithms, the contemporary emphasizes developing them and choosing intelligently among different algorithms for the same task.

We now discuss the components of algorithmics in more detail. It is standard to divide algorithmics into three parts, design, verification, and analysis.

Algorithm Design is the process of algorithm creation. There are some general principles of algorithm design; it does not have to depend on unteachable flashes of originality.

The most important idea, as in much of mathematics, is to break a problem into pieces. If you can find a small building block that you understand, try to iterate on that block. To sum a sequence of numbers, reduce to the case of summing two numbers; create a running sum and add one more number to it each time. To multiply two large numbers (Example 1), figure out a way to reduce it to many instances of multiplying two one-digit numbers. To multiply two matrices (Example 3), first use the definition (2) to reduce this to many cases of a real-number calculation, and then use iteration to return this to single additions and single multiplications.

Sometimes one does not immediately see how to reduce a large problem to small pieces. Then one tries to reduce it to slightly smaller pieces. What is the gcd of two large numbers m and n (Example 2)? Well, does some slightly smaller pair of numbers have the same gcd? Yes, $m - n$ and n have the same gcd as m and n, because anything that divides (evenly into) m and n divides $m - n$ and n, and anything that divides $m - n$ and n divides $m = (m-n) + n$ and n. And if subtracting n from m once preserves the gcd, then subtracting as many times as possible, leaving the remainder when m is divided by n, also preserves the gcd. This is the insight that leads to Euclid's algorithm.

The algorithm for Towers of Hanoi is also based on reducing to a smaller case. You can solve the game with n rings if you can solve the game with $n - 1$ rings, as shown in Fig. 2.

There are, of course, many other principles of algorithm design, and whole university courses are devoted to it. Here we'll mention two more, ***top down design*** and ***bottom up design***. The former refers to outlining the big picture first, and then filling in the details of the parts later. The latter refers to starting with small pieces and putting them together to do the whole job. While top down is generally the better approach for involved problems, both approaches have their roles.

Algorithm design is more or less the same thing as problem-solving methodology. Since mathematics education is permeated with problem solving, algorithm design is rightly an important component of a modern mathematics education. Practice in design not only makes people more successful at solving problems, but also it results in algorithms that are easier to communicate to others and to verify.

Algorithm Verification is the process of confirming that algorithms solve the problems they claim to solve; in other words, proving algorithms correct. Since loops are a primary aspect of algorithms, and since a loop can be iterated any nonnegative integer number of times, mathematical induction is the key method of verification.

Take Euclid's algorithm (Example 2). Let $P(k)$ be the statement that, just before commencing the kth pass of the repeat loop, gcd(num, denom) is the same as the gcd of the original m and n. That $P(k)$ is true for all $k \geq 1$ is easily proved by induction, using the fact that $\gcd(m, n) = \gcd(n, r)$ where r is the remainder when m is divided by n. (We argued this fact informally when Example 2 was introduced, and again somewhat differently five paragraphs ago.) When the loop entrance condition is tested for the last time, denom = 0, and so clearly gcd(num, denom) = num, and num is the value output. So by the induction, the output equals $\gcd(m, n)$ and the algorithm is valid. A proof of correctness like this is called a proof by ***loop invariant***; the loop invariant is the statement you prove to be correct each time you enter the loop.

Or take the algorithm for Towers of Hanoi. We may do induction on n, using the statement that any call of procedure $\mathrm{H}(n,r,s)$ correctly moves n rings from pole r to pole s. Since the very definition of H involves itself with $n-1$ rings, induction is easy to carry out. In general, a recursive algorithm immediately suggests an inductive proof.

The specifics of how to do induction for algorithms is not the point here. The point is that induction is the right tool. Mathematical induction, heretofore regarded in some quarters as a specialized method for proving certain formulas for sums, must be viewed as a more central proof method in any curriculum that gives substantial emphasis to algorithmics.

Algorithms need to be verified because more and more our lives depend on them, and once they are in place (say in our bank, to maintain our account records) they tend to get treated as black boxes. To be honest, algorithms used in the world at large are very complicated, too complicated for humans to carry out detailed mathematical proofs of correctness; and machine verification of correctness is still in its childhood. Thus, empirical debugging techniques play a vital role.

But mathematical verification should not be dismissed. First, big programs use many small building blocks which can or have been verified. Second, the algorithms whose correctness you are primarily responsible for are the ones you create yourself, and knowledge of how to verify an algorithm can be helpful at the design stage. If you propose to include a loop in your algorithm, and you know that the way to validate it is with a loop invariant, you will devise the loop invariant before you write the loop, and then you can write it to be sure that the loop invariant is preserved.

Algorithm Analysis is the process of determining how long an algorithm takes to run, and comparing that run time to that of other algorithms for the same problem and to absolute standards for that problem. "Run time" is a rough way to put it, since that suggests an actual machine (or person) to perform it, and different machines (and persons) will perform differently on the same algorithm. Usually one picks some salient feature, say the number of real-number additions if addition is the main operation in the algorithm under consideration, and determines the number of repetitions of this feature as a function of the input size. This function is called the complexity of the algorithm, or its efficiency.

Take, for instance, our algorithm for matrix multiplication (Example 3). If the two input matrices are both $n \times n$, then there are n^2 entries to compute, and each entry requires n real-number multiplications and $n-1$ additions. Therefore, the whole algorithm takes n^3 steps (if only multiplications are counted), and $2n^3-n^2$ (if additions and multiplications are counted). Or take Towers of Hanoi. The obvious thing to count is number of ring moves. It turns out that, if there are n rings, the algorithm takes 2^n-1 moves. If t_n is the number of moves with n rings, the recursive definition of procedure H leads to the conditions

$$t_{n+1} = 2t_n + 1, \qquad t_1 = 1; \tag{6}$$

the unique solution of these conditions is $t_n = 2^n-1$.

Calculations like these become valuable if the number of steps appears large and one wonders whether the problem will be tractable with the computing equipment available. Suppose, for instance, that a problem requires $n!$ steps when there are n input data. (Brute force approaches to the famous Traveling Salesperson Problem take this many steps, and the best exact methods known are in general not much better.) Then when n is merely 25, a computer that could do a billion steps a second would still take 50 million years to solve the problem! In contrast, the same computer could play 25-ring Towers of Hanoi in only .003 seconds, and could compute the product of two 1000 x 1000 matrices in a second.

These efficiency calculations become even more interesting when you have more than one algorithm for the same problem. Take Example 9 for computing $a(b_1 + \cdots + b_n)$ on a hand calculator. The best approach we discussed takes $n+2+C$ button-pushes, where C is the number of pushes needed to enter all of $a, b_1, \ldots, b_n$; two others took $n+3+C$ and the fourth approach took much longer. On a hand calculator, each button-push takes time, so even a saving of one is significant. Furthermore, there are lots of other, elementary problems where different calculator methods make a considerable difference. Take the problem of evaluating a polynomial

$$p(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_0.$$

There are a great many multiplications involved, especially if you don't have an exponential key. But there is another way to write this polynomial, best understood in traditional notation if we use a numerical example. If

$$p(x) = 4x^4 + 3x^3 + 2x^2 + x + 8,$$

then in *nested form*

$$p(x) = x(x(x(4x+3)+2)+1)+8.$$

Calculating $p(x)$ in this form takes many fewer steps (count them) and it's easy to carry out even with a simple four-function calculator and no memory. By the way, to make clear how both approaches work in general, and to make the algorithms precise enough to count the steps without confusion, the first thing to do is figure out how to state them in algorithmic language.

Or take the algorithm for constructing $\sqrt{n}$ (Example 4). It takes $n-1$ right triangle constructions to obtain $\sqrt{n}$. Noting this surely will inspire students to find a better way.

For Euclid's algorithm, the number of steps depends on the specific input, not just the size (number of digits) of the input. And in the Two Heads algorithm (Example 8), there is no input at all, but the number of steps varies. In these cases one takes several measures of the algorithm's efficiency – best case, worst case and average case. Average case is especially important, but usually hard to analyze.

The hardest problem is to compare an algorithm to an absolute standard. The complexity of a *problem* (as opposed to the complexity of an algorithm) is defined to be the number of steps needed by the best possible algorithm for the problem. Problem complexity is the subject of much current research – it's hard to figure out the complexity if, as usual, you don't know what the best algorithm is. For instance, it is known that Arabic multiplication and standard matrix multiplication are *not* the best algorithms for their problems, at least when n is quite large, but no one knows what the best algorithms are or how fast they are. Nonetheless, progress has been made in finding bounds on problem complexity. And every once in a while the complexity of a problem can be determined completely. For instance, it is not hard to show that the algorithm we gave for Towers of Hanoi is optimal.

In closing this section, we note that there are other ways to analyze the goodness of an algorithm than speed. One can consider space complexity – how much storage is needed. One can also consider numerical stability. For instance, in solving a quadratic (Example 6), if $b > 0$ and $4ac$ is very small compared to b^2, then $(b-s)/2a$ is practically 0, and roundoff error may swamp the computation. In this case it is better to set x_2 to $2c/(b+s)$. Algebraically, the two formulas are equivalent, but less roundoff error is introduced in the latter since $b+s$ is not near 0.

3. Why Study Algorithmics?

We have already given our main reason: the use of sophisticated algorithms to solve problems is already pervasive in the world, and so informed citizens need to know what can be done by algorithms, how it is done, and how algorithms can be assessed. Also, a fair number of people need to know how to create algorithms.

Mathematically, this is an extrinsic justification; algorithms are important, so students ought to study them whether or not they are interesting mathematics or do good things for mathematics education. Fortunately, there is equally strong intrinsic justification.

First, introducing algorithmics in school raises fresh questions about old material and allows for greater student creativity. As Example 9 (calculator efficiency) shows, even basic arithmetic is no longer cut and dried. Too many traditional curricula consisted of many computational courses where students were told the right methods, and a few proof courses (say, classical geometry) where they were asked to be creative, but in a narrow theoretical way. In contrast, each question of the sort "devise an algorithm for ..." allows for many correct answers (not all equally good). Even a student who does not have a good theoretical grasp of the problem at hand may come up with a correct algorithm.

Even incorrect algorithms can have worthy features. They may involve good *heuristics* – imperfect but insightful ideas that often lead to a reasonably good solution in a reasonable amount of time. Also, an analysis of their flaws may be instructive and lead to interesting class discussions. For instance, suppose you want to pick a random set of two distinct numbers from 1 to 10. What's wrong with picking a number i at random from 1 to 9 and then picking a number j at random from $i+1$ to 10? I once heard a businessman say, speaking at a college graduation, that in the outside world one learns from one's failures. While there may not be much to learn from mistakes in traditional rote calculations, there is a great deal to learn from one's failures in devising algorithms.

A second intrinsic reason for studying algorithmics is: it can help students understand traditional mathematics better. You really have to understand a procedure well in order to "explain" it to a computer, or to write it in algorithmic language. For instance, to understand Arabic multiplication well enough to describe it in algorithmic language (Example 1), you really have to understand place notation and the distributive law. And it's not just procedures that come to be understood better, but abstract concepts as well. For instance, the function concept is concretized by seeing algorithms turn inputs into outputs. Real numbers are made more concrete as the student sees (Example 7) that successive rational approximations needed to compute them. Moreover, when students test their algorithmic representations by running them on computers, they get instant feedback as to whether their constructions are correct.

There are also arguments against studying algorithmics in school. The basic argument goes: the sort of questions emphasized in algorithmics are already outdated, or soon will be. For instance, algorithmics puts great emphasis on the relative efficiency of algorithms. But if one approach to a problem takes n^2 steps and another takes n steps, the difference in actual seconds will be unnoticeable for the values of n used in any classroom. Or, why bother discussing methods for doing computations on a four-function calculator when soon all calculators will be much more powerful? Indeed, what's the point of talking about the merits of different methods of polynomial evaluation, when on the now popular "algebraic" calculators, you can punch in the definition of a function as a formula in x, then punch in a numerical value for x and finally just hit the EVAL button? One doesn't need to know *any* method for breaking down the evaluation of the formula into small steps because the calculator does it all.

Generalizing, computing devices are getting more and more advanced in the sense that they can respond to higher and higher level commands. When you think that, in the wings, there are machines that will create proofs and create algorithms for solving problems, why do students need to be schooled in the ability to create algorithms themselves?

I answer as follows. No doubt the level at which it will be appropriate to do algorithmic analysis will change over time. I really like to discuss different methods of polynomial evaluation with my classes, but one day (perhaps soon) this may seem as outdated to them as if I were to explain the theory behind slide rules. But if we can draw any lesson from the history of computing technology, humankind, including students, will always use technology to its limits, and its most powerful use will always involve the interaction of human and machine. To pick a simple example, I am not worried that the difference between an n^2-step algorithm and an n-step algorithm will be lost on students. First, some student always tries to run a recently learned algorithm on data that is too large, and wonders out loud why the machine sat spinning its wheels. Second, even if most students stick to small "textbook" data sets, it is easy to show them that in the outside world some very large problems must be solved where differences in algorithm efficiency are crucial. Sorting and searching (discussed later) provide good examples; governments and large businesses must sort and search enormous data sets.

The issue, then, is to keep the algorithmic examples up to date. This can be done if educators keep informed about the latest research and the latest technology.

Sometimes the exact opposite reason is proposed for not studying algorithmics. It is a theorem that there is no algorithm for determining which problems are solvable by algorithms. (This is because the "universal Turing machine" cannot solve the "halting problem".) So to emphasize algorithmics either misleads them about what algorithms can do or cuts them off from problems that have no algorithmic solutions.

But we do not propose that only algorithmic approaches to mathematics be studied. We only propose that algorithmics receive much more attention than previously.

4. Suggestions For Implementation

Two disclaimers: First, my knowledge of curricula worldwide is limited, and so I speak mostly from an American viewpoint. Second, in a paper this length, one can at most give illustrative examples and broad ideas of how to implement algorithmics. For more detailed ideas, appropriate at least in North America, see [NCTM 1989, Kenney 1991]. The suggestions below concern the primary and secondary levels except for a few brief remarks about the university level at the end.

Look at traditional computations more closely. Basic arithmetic, computations with polynomials, solutions of linear equations – such things are often taken as routine and devoid of opportunity for fresh thought. But from the viewpoint of algorithmics there is plenty to think about. Students can discover traditional algorithms using design principles, and discover alternative algorithms. While they are unlikely to discover significantly faster algorithms, they can be told (or, at a higher level, shown) that faster algorithms exist, and that best algorithms are unknown.

Treat nontraditional computations related to classical questions. In every country students learn closed form solutions to certain sorts of equations, but they don't always look closely at how to evaluate those solutions accurately, or discuss methods for approximately solving equations without solution formulas. Students often learn to count permutations and combinations, but they don't often consider how to efficiently list all of them of a certain size, or generate a random one. In short, classical formulas that don't appear algorithmic raise algorithmic issues.

Introduce some new topics. There are whole fields of mathematics, with many applications, that have an algorithmic flavor and are not represented at all in many curricula. Many of these are grouped these days under the headings ***discrete mathematics***, ***operations research*** and ***theoretical computer science***. Here are a few examples, but at this point they are little more than name-dropping, and one should refer to texts in these fields, such as [Hillier and Lieberman 1986, Manber 1989, Maurer and Ralston 1991].

Difference equations is the study of inductively defined sequences such as the step-count sequence t_n of Towers of Hanoi in Display (6). These include the traditional arithmetic and geometric sequences and series – and much more. Computing terms in inductively defined sequences is immediately an algorithmic question, and conversely, analyzing algorithms reduces to analyzing difference equations.

Graph theory, in the sense of networks, is full of algorithmic questions. If a graph represents an existing road network, how do you find the shortest routes between points (in distance, time, or whatever)? If the network represents the possible links between cities is a telephone network planned for a developing region, how do you decide which set of links will connect up the region at minimum cost? There are a variety of good (and not so good) algorithms for such problems, and many of these algorithms are not hard for students to discover.

Sorting and searching (e.g., alphabetizing and looking through an alphabetized list for a word) are standard computer science examples that wouldn't traditionally be thought of as having any mathematical content – clearly it is possible to sort and search, so what's the problem? But once again, there are lots of different methods, with various efficiencies, and various challenges to verify them and analyze them.

Make computing power available to students. This is a tall order. No matter how rich the country, there are always newer and more powerful devices one could want, and even in rich countries it may be a long time before there is one computer per student in every class. But the point is, algorithmic questions take on much more life when students have what they regard as powerful computing aids, and then they discover they can still devise problems that aren't solved instantaneously. As discussed earlier, even four-function calculators are very helpful in bringing to life algorithm design and efficiency questions. With computers as well as calculators, one can start in the early years with such things as the language Logo and Turtle graphics, and move in later years to computer algebra systems.

Introduce algorithmic language. Whatever computing power is available, precise methods for describing algorithms are necessary if algorithms are to be an object of study and not just something students perform. There is no standard algorithmic language, and perhaps different sorts of languages are best for problems to be treated with different sorts of machines (or by hand). Nonetheless, it is not hard to devise useful language constructs.

Put more emphasis on mathematical induction. We have indicated how induction is the main method for validating algorithms. Actually, induction can be viewed more broadly, and as such is at the foundation of algorithmics. There are inductive discovery techniques (reduce to the previous case, or build up from small cases to find a pattern), inductive definitions, as in Display (6), inductive algorithm commands (loops and recursive procedures) as well as inductive proofs.

Eliminate the schism between solving and computing. Traditionally there is pure mathematics and applied mathematics. Pure mathematicians prove that solutions exists, and applied mathematicians figure out how to find them. In algorithmic mathematics, good computation methods are found simultaneously with showing that solutions exist. By putting these two issues together right from the earliest years, we help to overcome what has sometimes been an unfortunate two-class system in mathematics and science.

A few words about the university level. Here the schism between pure and applied has been particularly acute. But it is breaking down. Many research mathematicians in pure fields are finding algorithmic questions interesting. Some algebraists, for instance, are now very interested in how classical objects in group theory can best be computed [Beeson 1990; Mines, Richman and Ruitenberg 1988]. This could filter into the classroom. Even in calculus, some questions can be given a much more algorithmic flavor than they have been. The rules of differentiation, instead of simply being a set of rules, can be viewed as the parts of an algorithm that determines the derivative for any elementary function (once elementary functions are given an inductive definition!). The rules of integration can be viewed as part of an algorithm that determines the integral of some elementary functions, and some discussion can be added that integration is no longer an "art", because there is an algorithm for determining exactly when a function can be integrated in closed form. Those university mathematicians who have gotten interested in algorithmic questions should be encouraged to share with their colleagues their ideas about how to introduce these new approaches in the standard courses.

REFERENCES

Beeson, M. [1990]: Review of "A Course in Constructive Algebra", *Amer. Math. Monthly*, 97 (April), pp. 357–362. (See also Mines below.)

Hirsch, C. and Zweng, M. [1985]: *The Secondary School Mathematics Curriculum*, (1985 NCTM Yearbook), Reston, VA: National Council of Teachers of Mathematics.

Hillier, F. and Lieberman, G. [1986]: *Introduction to Operations Research*, 4th ed., Oakland, CA: Holden-Day.

Kenney, M., (ed.) [1991]: *Discrete Mathematics Across the Curriculum, K–12* (1991 NCTM Yearbook), Reston, VA: National Council of Teachers of Mathematics.

Malkevitch, J. et al. [1988]: *For All Practical Purpose*, San Francisco: W. H. Freeman. Also available as 26 videotaped TV programs.

Manber, U. [1989]: *Introduction to Algorithms, A Creative Approach*, Reading, MA: Addison-Wesley.

Maurer, S. [1984]: Two meanings of algorithmic mathematics, *Mathematics Teacher*, 77 (September) pp. 430–435.

Maurer, S. [1985]: The algorithmic way of life is best, *College Math. J.*, 16 (January) pp. 2–18 (Forum article and reply to responses).

Maurer, S. [1991]: Proofs and algorithms: a reply to Gerstein, *UME Trends*, Vol. 3 (January) p. 8.

Maurer, S. and Ralston, A. [1991]: *Discrete Algorithmic Mathematics*, Reading MA: Addison-Wesley.

Mines, B., Richman, F. and Ruitenberg, W. [1988]: *A Course in Constructive Algebra*, New York: Springer-Verlag.

NCTM [1989]: *Curriculum and Evaluation Standards for School Mathematics*, Reston, VA: National Council of Teachers of Mathematics.

North Carolina School of Science and Mathematics (Barrett, G. et al) [1991]: *Contemporary Precalculus through Applications*, Providence, RI: Janson Publications.

Peressini, A. et al. [1992]: *Precalculus and Discrete Mathematics*, University of Chicago School Mathematics Project, Glenview, IL: Scott, Foresman.

Ralston, A. [1981]: Computer science, mathematics and the undergraduate curricula in both, *Amer. Math. Monthly*, 88, pp. 472–85.

Ralston, A. [1984]: Will discrete math surpass calculus in importance? *College Math. J.*, 15 (November) pp. 371–382 (Forum article and responses).

Ralston, A. and Young, G. S., (eds.) [1983]: *The Future of College Mathematics: Proceedings of a Conference/Workshop on the First Two Years of College Mathematics*, New York: Springer-Verlag.

ON THE MATHEMATICAL BASIS OF COMPUTER SCIENCE

Jacques Stern
Équipe de logique, Université Paris
et
Département de Mathématiques et Informatique, École Normale Supérieure
France

It is now clear to an anybody that a working mathematician cannot ignore computers: as a consequence, it is commonly admitted that students in mathematics, and especially those who intend to be teachers in the field, have to be exposed to some high-level language (such as Pascal). Nevertheless, this is far from enough: the question of whether students in mathematics should be familiar with some parts of the theoretical foundations of computer science cannot be avoided because these topics are precisely the parts of computer science close to mathematics and seem to be necessary in order to establish connections between both fields that go beyond the ability of using the computing power of modern machines.

In France, following this line of ideas, the study of algorithms and related topics has become, in most universities, a significant part of the standard curriculum leading to graduation in mathematics. Also, an optional test in computer science has been offered for a few years in the well-established "Concours d'Agrégation de Mathématiques", which is a kind of "teaching Ph-D", passed by most of the teachers for the age-group 17-22.

The author has recently published a book entitled "Fondements Mathématiques de l'Informatique"[1990], which covers a large part of the requirements in computer science for undergraduate programs in mathematics. The aim of the present contribution is precisely to present some general ideas that grew during the process of writing up that book. These ideas are my personal views although I owe a great debt to many colleagues with whom I have had inspiring discussions.

Before going into greater detail, let me make one remark: Mastering some of the basic tools in computer science will not turn a mathematician into a computer scientist. Instead, it should help to develop a different frame of mind, suitable to understand the specific features of computer science. This is most important for a mathematician because, as is shown in other contributions in this book, these specific features will necessalily affect both the teaching and the practice of mathematics themselves.

Around the notion of computation

Computation Theory is considered by many people to be a very dull subject; nevertheless, it is the first burden of the theory to provide a suitable criterion for drawing a limit between what is *computable* (or *effective*) and what is not. A simple way would be to use the word *computable* for everything that can be processed on a real computer. Although this point of view is not completely meaningless, it remains rather vague and cannot be considered as a genuine mathematical notion because of its lack of precision. Furthermore, this point of view is not even historically correct: a lot of outstanding work connected with the subject of computation theory was published before the first modern computer was built. For example, note the work of Turing [1936], Post [1936] on computation theory itself, and also the work of McCulloch and Pitts [1943] on the modelling of neuron nets, from which the theory of automata grew.

It is precisely the theory of automata that we we propose to choose as a starting point. Many reasons can be put forward in order to justify such a choice. The theory is simple, established on firm mathematical grounds and provides various exercises in programming: for example, one can simulate an automaton in a high-level language like Pascal or discuss algorithms that compute the minimal automaton. Also, the concept of non-determinism, which is of utmost importance in theoretical computer science, can be quickly and naturally introduced in a simple setting. Finally, the theory of automata has several applications: to text editors and compilers in particular; this is not a minor argument.

Nevertheless, one can easily come to the conclusion that automata do not provide a satisfactory model for real machines. This conclusion can be reached by writing down simple languages that are not accepted by a finite automaton but also through the convincing observation that a central feature of computers is completely wiped out, namely their ability to store data in a memory. We are thus back to our original problem of defining the notion of *computable* and it is reasonable, at this point, to require that this notion should be described using various different techniques that come out to be equivalent: this will ensure that a mathematical invariant has really been found and this will make Church's Thesis highly plausible. (Recall that Church's thesis states that the notion of machine-

computable function and the mathematical family of *recursive* functions are identical).

Four distinct approaches can be taken.

- **Adding a memory device to a finite automaton.** This yields the definition of a Turing machine.
- **Directly modelling actual computers.** This can be done through the notion of a random access machine (cf. Cook and Reckhow [1973]) operated by a very simple language similar to machine code.
- **Defining a simple class of programs.** For example one can define a restricted version of Pascal which uses only the integer type and the control sequences **if** ... **then** ... **else** and **while** ... **do**.
- **Defining the class of (partial) recursive functions.** This is a good opportunity to discuss functional languages: recursive definitions can be handled by using constructs that are exactly similar to those appearing in Lisp.

The proof that all these definitions are actually equivalent is a source of very interesting observations. For example, the fact that the restricted version of Pascal can compute all recursive functions proves the well-known fact that the **goto** statement can be dispensed with. It may be worthwhile to note that replacing **while** ... **do** by **for** only allows the computation of primitive recursive functions. Also, the simulation of a random access machine by a Turing machine is a good exercise that shows how to handle a sequential memory.

Once the notion of a computable function has been given a precise definition, it becomes possible to discuss decidability issues: By coding Turing machines and constructing a universal machine, it does not require much more effort to state correctly the "halting problem" and show that it is semidecidable but not decidable (which means that a machine can find positive answers in a finite computation time but cannot do the same both for positive and negative answers). It is not clear that the study of *general recursion theory* should be pursued. Still, one may wish to present the semantics of recursive procedures and the fixed-point approach to programs and develop the recursion-theoretic tools that are needed, such as Kleene's theorem (which basically states that the name of a recursive function can be used within its own definition).

Then, one can have a discussion on whether or not the dichotomy decidable/undecidable is of practical significance. This is a way to introduce Complexity Theory through the constraints of time. Going back to the various mathematical models of computation, one can explain how a basic *cost* can be attached to the execution of each instruction, the overall cost (or *complexity*) being the sum of all basic costs. Thus, one can define the *complexity function* of an algorithm which measures its cost in terms of the size of the data. Of course this complexity depends on the abstract machine chosen but one can check that, when one machine is simulated by another, the complexity functions are polynomially related. This allows the definition of the class $\mathcal{P}$ of polynomial time computations, which is a reasonable candidate for modelling a class of problems sometimes called *feasible* or *tractable*.

Around the notion of algorithm

Now that we are equipped with a theoretical notion of complexity, it is necessary to use it in concrete situations. This can be done through a review of various algorithms. This review is, by no means, an exercise in programming style, even if correct programs have to be written at some point. The emphasis should be on the design and analysis of algorithms, which are very closely connected. Of course, the rules of the game should be clearly stated and discussed, especially the choice between the two main notions of complexity that are in use: worst-case analysis and average-case analysis. This choice depends on the underlying model: for example, average-case analysis is relevant when the probability of "ill-behaved" cases is small. In both cases, the analysis is combinatorial in character and quite often yields non-trivial recurrence relations. In order to handle these, some specific tools are needed, like the statistics of permutations and distributions and the use of generating series (cf. Knuth [1973]). Generally, such techniques (e.g. the use of singular points of the generating series) only allow an asymptotic analysis and one may ask if this kind of information has any practical meaning: after all, the size of the data are bounded by the computing environment! It turns out that the asymptotic analysis is actually relevant: When a given algorithm runs in time $O(n \log n)$, for example, it is usually true that the constant implicit in the O notation is rather small and that the asymptotic behaviour is reached rather quickly.

The students should also get used to performing the analysis of the complexity of an algorithm without going back to the original definitions, based on abstract models of computation. If the size of the integers is bounded (which is often the case in practical situations), the complexity is roughly the number of machine instructions performed during execution. This validates the use of the overall number of comparisons as a measure of complexity for sorting algorithms. When large integers are involved, things become a bit more complicated: a convenient way is to multiply the number of instructions performed by n^2, where n is the number of digits of the integers used. This is to take into account the cost of multiplication as $O(n^2)$.

Together with algorithms the specific data structures used in computer science should be discussed: stacks, files, trees, graphs etc. It should be stressed that this point of view is quite different from the one that was taken in the previous section: In computation theory, we considered simulations involving basic manipulations on data structures and we claimed that these manipulations were not costly, becauce we were interested in the general notion of polynomial time. In practical cases, a given polynomial time algorithm can be superior to another one and, very often, the choice of a good data structure may actually save a significant part of the running time.

The choice of algorithms that can be reviewed is quite large and depends on the mathematical background of the students. At an advanced level, it is probably more rewarding to give examples that use mathematics in a non-trivial way, such as:

- **The fast Fourier transform** and its application to fast multiplication of integers (in time $O(n \log n \log\log n)$).
- **Basic algorithms for computer algebra**. This can be an opportunity to demonstrate the use of a computer algebra system, like Maple or Macsyma.
- **The simplex algorithm** for linear programming.
- **Primality tests**, at least probabilistic ones.

Unfortunately, it is not possible to discuss these algorithms in detail. and we will only briefly comment on the last example. As is well known, testing primality by sieving requires a large amount of time and memory. In order to overcome this difficulty, one may try to use the mathematical properties of prime numbers. For example, it is known that, whenever p is prime and a is not zero modulo p, the so-called Jacobi symbol $\left(\frac{a}{p}\right)$ is equal to $a^{\frac{p-1}{2}}$. This is not the case in general. More precisely, if n is not prime, at most one half of the possible a's satisfy the equality

$$\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \pmod{n}.$$

As was observed by Solovay and Strassen, the computations required to compute Jacobi symbols and exponentials modulo n can be performed efficiently. This makes it possible to recognize whether or not a given integer n is prime by picking random values of a and testing the above equality. If sufficiently many tests are successful, n is declared to be prime.

At a more elementary level, examples can be taken from the following list (which is not exhaustive):

- **Sorting**. This should include a comparison of various algorithms and a discussion of *quicksort*, as an illustration of the power of the *divide and conquer* method.
- **Searching**, with an emphasis on the choice of specific trees as data structures.
- **Pattern matching**, because of the connection with automata.
- **Graph algorithms**, for the nice interplay between discrete mathematics and computer science.

Graph algorithms can be a way to introduce $\mathcal{NP}$-complete problems. Indead, one can observe that computing shortest paths can be done in a very efficient way whereas no polynomial time algorithm is known for many graph problems, such as the *Hamiltonian path problem*. This problem can be described in very concrete terms as follows: Given a set of cities together with possible air connections between them, can one tour all the cities, visiting each city once and returning to one's starting point? In order to handle this problem, one can

- guess a plausible solution
- check its correctness (in polynomial time)

All problems that can be solved in such a nondeterministic manner are called $\mathcal{NP}$-problems, and an $\mathcal{NP}$-complete problem is an $\mathcal{NP}$-problem that can be used as a "subroutine" in order to solve all other $\mathcal{NP}$-problems, with polynomially many extra steps of computations. It is an open problem (probably the most important problem in theoretical computer science) whether or not an $\mathcal{NP}$-complete problem can be solved through a polynomial time algorithm, and as a consequence, $\mathcal{NP}$-complete problems are considered to be difficult: They can only be attacked by time-consuming techniques such as backtracking.

Of course, the class of $\mathcal{NP}$-complete problems can be given a formal definition through nondeterministic Turing machines. Once this is done, one can prove Cook's Theorem (Cook [1971]), stating that the *satisfiability problem* for clauses of the propositional calculus is $\mathcal{NP}$-complete. More examples of $\mathcal{NP}$-complete problems can be given (cf. Garey and Johnson [1979]), such as:

- *The travelling salesman problem*
- *The knapsack problem*
- *The clique problem*

Finally, some indications can be given on how to handle $\mathcal{NP}$-complete problems and also on practical applications of these notions, through the use of "one-way" functions.

Around logic

Many authors now emphasize the role of logic in the foundations of computer science. This is presumably because of the deep connection that exists between computer programs and proofs. This connection was already implicit in the section on computation theory: The undecidability phenomenon is closely related to Gödel's Incompleteness Theorems that show the extreme limits of deductive mathematics.

It is therefore necessary to include a thorough introduction to logic in order to endow mathematicians with a synthetic view of computer science. But it should be added that the interplay between mathematical logic and computer science is such that logic cannot be taught now as it it was before the advent of computers. This applies both to the formal presentation of syntactical objects and to the development of the theory itself.

From the formal point of view, it is extremely helpful to follow computer scientists and to consider formulas as trees and not only as strings of symbols, as was done classically. With this approach, a notion such as *a free occurrence of a variable* is given a clear, almost geometrical definition, which was not the case when it was introduced through a cumbersome recurrence. This can be quite important considering the fact that syntax must be quickly understood by students who have *not* been exposed to logic beforehand.

For the same kind of reasons, students have to be motivated as early as possible. Indeed, this can be done by discussing the aim of artificial intelligence: How to make correct inferences from a database of known facts. This is meaningful even in the simple framework of propositional calculus and the difficulty of the problem can be understood by recalling that the satisfiability problem is $\mathcal{NP}$-complete. The search for solutions to the deduction problem that are not brute search algorithms leads to the method of resolution, which can be made very efficient in the particular case of Horn clauses through linear resolution. For the convenience of the reader, let us recall that *clauses* are disjunctions of *literals*; literals are either *positive*, i.e. propositional variables or *negative* (negation of such variables). Horn clauses include at most one positive literal. The resolution method is a way to derive a contradiction from a set of clauses by making systematic use of the tautology

$$(p \vee q) \wedge (\neg p \vee r) \to q \vee r.$$

As far as the predicate calculus is concerned, it is almost compulsory to use a constructive approach based on Skolem functions and Herbrand's theorem. (Recall that Skolem functions ensure that, whenever a formula $\exists x \Phi(x, y_1, \cdots, y_n)$ holds, a possible solution x of this can be computed by a term $f(y_1, \cdots, y_n)$.) Herbrand's theorem states that, provided Skolem functions exist, any set of formulas from which no contradiction can be derived can be realized in a model whose domain is the set of closed terms). This provides both completeness and compactness by reduction to the propositional calculus. At this point, one should not avoid discussing undecidability issues again: even if one starts with a *finite* set of formulas, one usually gets an infinite number of Herbrand clauses and therefore the Herbrand procedure does not necessarily come to a stop.

In the above setting, the search for a more efficient procedure leads to Robinson's *unification-resolution* algorithm, and as in the case of the propositional calculus, one has to restrict oneself to Horn clauses if one is really concerned with efficiency. As is well known, such a restriction enables the use of backtracking and this is basically the strategy of the Prolog language. The study of Prolog offers a very interesting application of logic in computer science. It shows that the views of artificial intelligence can be turned into an actual programming methodology. Of course, it is clear that Prolog is a programming language and *not* a theorem prover and that completeness is lost, as a consequence of various features of the actual language, as the lack of the so-called "occur-check" and the use of the "cut" primitive. In order to show how the language works, simple programs can be written and discussed.

Now, Prolog is not the only example of application of logic to computer science and one can choose to give an exposition of program verification through Hoare's logic. Recall that this method is based on cutting the execution path of a Pascal-like program into loop-free pieces. To each cutpoint A is attached a formula ϕ_A, whose free variables are the actual variables of the program. Logic comes into the picture in proving that, if execution leads from A to B and if ϕ_A is true at A, with the current values of the variables, then ϕ_B is true at B, with the resulting values of the variables. This is used to show *partial correctness* of the program, which means that, if execution terminates, the final formula expresses that the result is as expected. *Total correctness* can be proved along the same lines by using a well-founded relation and proving that loops decrease values of the variables with respect to this well-ordering.

Finally, another topic where logic and computer science interact at the conceptual level is the λ-calculus, considered as another approach to program-correcness. Once again, it is based on the connection between algorithms and proofs. This time one is talking about formalized proofs within the framework of intuistionistic logic (without use of the middle-third) and about systems of rules using the typed-λ-calculus, where a proof yields a term t, which is, in a way, its algorithmic content (cf Krivine and Parigot [1990]). In order to be more precise, let us recall that λ-calculus builds *terms* from variables, through the following rules:

- if t and u are terms, then (tu) is also (application of t to u)
- if t is a term and x a variable, then $\lambda x.t$ also (abstraction)

The λ-calculus can be considered as a kind of machine language, a term being turned into a so-called normal form by reduction rules. In order to program a function with integer arguments (for example), one proves a formula (stating that the result is an integer). This gives a term t and execution is just the reduction of the application of t to the terms denoting the arguments. Because of the way programming is performed, correctness is ensured. Of course, the work on this type of programming strategy is only beginning and one should not conceal that the resulting programming style is highly inefficient at this stage.

More on syntax

Because of the organization of our paper around computation, algorithms and logic, we have not discussed some quite interesting connections where mathematics provides the necessary background. For example, we mentioned that logical formulas can be considered as trees and the same is true of computer programs. Now both usually appear as strings of symbols. It is therefore very important to be able to recover the full tree structure from its string version. This is a part of *compilation*, called *syntax analysis* (cf Aho, Sethi, Ullman [1986]). It turns out that the theory of *context-free* languages is exactly the tool needed to perform syntax analysis efficiently.

Conclusion

In this short paper, we have tried to describe what we consider as the mathematical basis of computer science, to show how the chosen topics can be organized and to motivate the choices that we have made. Following the further developments of computer science, these contents will presumably have to be expanded or modified. For example, it may appear important to discuss boolean networks (to model VLSI) or to introduce tools for the study of relational databases. In any case, we feel that mathematical tools for computer science will become a part of any advanced curriculum in mathematics.

REFERENCES

Aho, A.V., Hopcroft, J.E. and Ullman, J.D. [1983]: *Data Structures and Algorithms*, Reading, MA: Addison Wesley.

Aho, A.V., Sethi, R. and Ullman, J.D. [1986]: *Compilers*, Addison Wesley, Reading, Mass.

Clocksin, W.F. and Mellish, C.S. [1981]: *Programming in Prolog*, Berlin: Springer-Verlag.

Cook, S.A. [1971]: The complexity of theorem proving procedures, *Proc. 3rd Annual Symposium on the Theory of Computing*, 29-33.

Cook, S.A. and Reckhow, R.A. [1973]: Time bounded random access machines, *J. Computer and Systems Science* 7, 354-375.

Garey, M.R. and Johnson, D.S. [1979]: *Computers and Intractability, A Guide to the Theory of NP-Completeness*, San Francisco: Freeman.

Hoare, C.A. [1969]: An axiomatic basis of computer programming, *Comm. ACM* **12**, 576-580.

Kleene, S.C. [1939]: General recursive functions of natural numbers, *Math. Annalen*, 112, 727-742.

Knuth, D.E. [1973]: *The Art of Computer Programming: vol 3: Sorting and Searching*, Reading, MA: Addison-Wesley.

Krivine, J.L. and Parigot, M. [1990]: Programming with proofs, *J. Inf. Process. Cybern.*, EIK 26, 149-167.

Manna, Z. [1964]: *Mathematical Theory of Computation*, New York: McGraw-Hill.

McCulloch, W.S. and Pitts, W. [1943]: A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophysics*, 5, 113-115.

Post, E. [1936]: Finite combinatory processes formulation I, *J. Symb. Logic*, 1, 103-105.

Robinson, J.A. [1965]: A machine oriented logic based on the resolution principle, *J. ACM*, 12, 23-41.

Rogers, H. Jr. [1967]: *Theory of Recursive Functions and Effective Computability*, New York: McGraw-Hill.

Sedgewick, R. [1983]: *Algorithms*, Reading, MA: Addison-Wesley.

Stern, J. [1990]: *Fondements Mathématiques de l'Informatique*, Paris: McGraw-Hill.

Wirth, N. [1986]: *Algorithms and Data Structures*, Englewood Cliffs, NJ: Prentice-Hall.

THE EFFECT OF COMPUTERS ON THE SCHOOL MATHEMATICS CURRICULUM

Klaus-Dieter Graf
Freie Universität Berlin, Germany
Rosemary Fraser
University of Nottingham, U.K.
Leo H. Klingen
Helmholtz-Gymnasium, Bonn, Germany
Jan Stewart
University of Nottingham, U.K.
Bernard Winkelmann
Universität Bielefeld, Germany

Introduction

Historical Sketch and Trends

The three traditional cultural techniques (Kulturtechniken), which play the most important role in our children's education are reading, writing and calculating. From the time of their "definition" (perhaps 1200 years ago; Alkuin, an adviser of Charlemagne, mentioned them) the sets of methods establishing these techniques have undergone great changes and so did the subsets which were accessible at school levels. In our times the largest expansion occurred in calculating, which developed into a technique of solving problems formally with numbers, symbols, graphics and words. On one side, this is a result of extensive mathematical research, which among other results brought about powerful algorithms, easy to execute. On the other side this trend was accelerated by the rise of powerful processors for algorithms, namely computer systems together with their scientific background, informatics (i.e. computer science). These aids make a variety of formal problem-solving methods accessible for school mathematics and other subjects, which previously could not be executed by students and pupils. Algorithms form one important class of these methods.

The development outlined above caused and still has a significant impact on school mathematics education. At least three of the didactical dimensions of the mathematics classroom are envolved: content, method and medium, to say nothing of the pupil - teacher relationship. Control on these impacts can only be gained by integrating and organising them into mathematics curriculum at all levels, since, as A. Ralston [1990] points out " .. only .. curriculum content can serve as a lever to change the entire mathematics education system". Computer use in mathematics education started as a very special method with mostly special topics. Future computer use should be a standard method, applied in whole strands of subject matter. This article will give a review of some effective and successful steps and some reasonable trends in the pursuit of this goal in school mathematics.

In addition, many of the examples of this paper indicate that the technology is already a significant factor in school classrooms, a factor that more than deserves its place. The contribution that it can make to the social and academic interactions is vivid and, once experienced, always valued.

Finally, just as children play out a wide range of roles in being part of the community they are in, so too can computers. Thus we ask the reader to consider the computer as a member of the classroom community, one that is able to contribute to the day's activities in an appropriate fashion.

Considerations and concrete suggestions for the use of computers in mathematics teaching depend on knowledge about and experience with such instruments shared by teachers and mathematics educators. Fifteen years ago these people had access to computers mostly as programmers in numerically-oriented languages. So computing power was mainly used in secondary math education for numerical algorithms in the form of short Basic programs. Ten years ago, another step – but still in the algorithmic spirit – was taken with Logo on various home computers with its underlying philosophy of exploring mathematics in specially designed microworlds and of learning mathematics by teaching it to the computer; Logo also included the use of geometry and symbolic manipulations. Primary education was involved with these ideas, even kindergarten.

The proliferation of so-called standard software on personal computers in the last decade gave way to new considerations and experiments, especially with spreadsheets, programs for data representation, statistical and numerical packages, databases, CAD (Computer Aided Design)-software and computer algebra systems. But in the beginning such software was not very user-friendly, and afterwards became too complex; the need soon became obvious for special school adaptations which allowed easy specializations, employed mathematical notation similar to that used at school, and used powerful and helpful metaphors, so that even users with little training and only occasional practice (as is typical of school users) could succesfully handle them. This led to the creation of general and didactical software tools which sometimes also had a tutorial component, thereby integrating some traditions of computer-aided instruction (CAI). All these forms of using the computer came into being in sequence but can now be found simultaneously in discussions about mathematics teaching.

Even if suitable hardware and software are now available for ordinary schools, several necessary ingredients are still missing: Teacher training is far from sufficient; hardware availability in most schools is still dictated by the needs of computer science and computer awareness courses and the concentration of machines in special locations prevents or makes difficult the natural, selective use of software - e.g. a function plotter - during short episodes in the teaching process.

Influences on the Goals and Aims of Mathematics Teaching

In elementary schools children meet basic processes with patterns and numbers in the mathematics classroom for the first time. There is a range of uses of technology that have proved positive and stimulating in helping children to express themselves and to progress in a confident and enjoyable fashion. In particular these can help to discovery – partly unconsciously – of the importance of underlying structures as an aid to qualified communication in language and problem solving. The computer is well-suited to setting up structures - this will be illustrated in the examples that are discussed in detail in the section on Illustrative Software below. (For a more comprehensive discussion of the influence of computers on mathematics teaching, see the survey by Fey, 1989.)

The emergence of multimedia technology means that our communication with computers and, indeed, amongst ourselves will employ words, pictures and sound in equal partnership and will not be limited to a fixed sequential presentation. Although this article draws on the experience of using microcomputers in the classroom, it will also be relevant to the more sophisticated interactive video delivery that is now available.

At the secondary levels we consider two main aspects which influence the goals and aims of mathematics education: the (mathematical) preparation of students for their lives and occupations, and the role of mathematics and its applications in society.

The students' preparation for their lives and occupations starts in the first instance at school with its various disciplines. Since through the availability of computers, there are now strong tendencies to introduce simulations into the school teaching of science, most notably in biology, or of introducing elements of statistics and data analysis into the measuring sciences and geography (cf. Winkelmann, 1987), this is obviously a challenge to the teaching of mathematics: Mathematics should elucidate the principles, possibilities and possible pitfalls of these methods; ad-hoc-explanations of such methods by the specific content-oriented disciplines are surely not appropriate for giving the student a coherent appreciation.

It is important to realize that routine calculations of all complexities will be done increasingly by ubiquitously available machines which must be controlled at various levels by the users concerned. This requires more insight, more breadth, more ability to check consistency, but fewer routine algorithms. Such an emphasis belongs to the perennial goals of mathematics teaching, of course, especially in the new math movement. But now there is really the possibility of leaving out some of the drill because technology can take over. Even an insight into the fundamentals of computers and their programs may belong to the preparation for life. This can often be shared with the other formal discipline, informatics/computer science, if it is implemented. It is hard to be more specific, since the determination of the elementary and more advanced cultural techniques which are needed by the future citizens presupposes a futurist view of society which is notoriously hard to specify.

As to preparation for vocations, for university studies, fundamental ideas and experiences in algebra, geometry and fractals, analysis, data analysis and statistics, simulation and chaos would now seem to be necessary in different kinds of studies. More specific preparations for special vocations are again difficult to determine. For example, CAD (Computer-Aided Design which helps the construction of planar, spatial and other objects on the computer screen) is necessary for an increasing number of technical vocations, and this means the need for new and different qualifications in geometry; but what is exactly needed and how to build a curriculum to fulfill the needs of the trades remains unclear.

The same is also true for the other domains mentioned in this chapter; therefore, it is not laziness that the descriptions above are so general and unspecific. The general direction of necessary change can clearly be seen, but concrete decisions cannot be built on scientific knowledge yet; we have to experiment and gather ideas, examples and proven results in concrete circumstances.

Mathematics education at school not only has the task of delivering to students the qualifications asked for in vocations and daily life, but it should also give insight into the role of mathematics in culture and society, into the fundamental possibilities for understanding and description offered by mathematics, and into connected assumptions and limitations. In this respect, on the one hand today the greater part of the applications of mathematics is transmitted by the computer and thereby influenced in its character, as will be discussed in some instance below, and on the other hand the computer is fundamentally a mathematical machine and thus its proliferation is a tremendous amplification of the mathematization of our lives.

Primary School

Computers and Calculators for Young Children

The greatest impact of computers on the learning of school mathematics has occurred in secondary school. However, we wish to begin by discussing the primary school curriculum for three reasons:

- a natural and basically positive attitude towards computers can only be achieved at this level.
- since primary school determines a student's lifelong attitude toward mathematics, we must use all possible means - and the computer is one of the most powerful of these - to create a positive attitude during primary education.
- it is necessary that teachers planning to use computers in secondary school and even in universities understand what was done in primary school and what the problems were there.

The first major need to socialise with peer groups and to share them arises when children move out of the home into regular contact with others at playschool or infant school. Here, also, serious work starts in developing spoken and written language skills, learning about the world and meeting basic processes with patterns and numbers. Plenty of play and creative opportunities are provided to allow natural skills to flourish.

How can technology help in this busy active happy environment of early childhood? Technology is certainly part of the world that the children will grow up in but one might feel it is not yet a part that children need to meet directly. Indeed, there are concerns expressed in some countries that it might be positively harmful to allow the use of technology before certain basic skills have been mastered.

In the next section we shall look at some examples of use under 'content' headings although they also give rise to cross-curricula work. For ease of illustration we shall take Language Development, Early Science and Basic Mathematics as our main categories. The decision not to limit the primary school part of this article to mathematics is deliberate in view of the fact that most elementary school teachers carry a responsibility for the major part of a total curriculum. It is thus important that the use of computers be set in this context. However, the Language and Science examples also have a relevance to mathematical processes although this is not made explicit.

Before looking at the specific examples, it is necessary to discuss the social situation that children find themselves in. Basically, there is a teacher to whom they can turn and who organises their activities during the day; there is a group of children that they work with, those they play with plus special friends that they confide in. Thus children contribute to a whole range of interactions sometimes as part of a large class, at other times with a smaller group, often just to one other person and, finally, they must frequently work things out as an individual. In short, the challenge that young children face of being a member of the classroom community is complex and demanding.

Children need to develop good productive relationships and for this they need effective verbal and nonverbal skills. Communication through body language and other nonverbal signals develop naturally and requires no formal intervention. With the spoken and written word the structure of the language, although not formally expressed, begins to be unconsciously absorbed and then actively used to build new sentences and expressions. This somewhat surprising occurance indicates the importance of underlying structures as an aid to communication. The possible role of the computer in this process was mentioned above.

We shall analyse, albeit in a rather crude fashion, the roles played out by teachers, children and computers in the examples that follow.

Thus the focus of the following descriptions will be to consider the quality of the communication in the classroom community and to identify structures and roles that enhance the interactions between members of the community.

Illustrative Software

1. Language Development

We describe here an extremely simple but powerful program called DEVELOPING TRAY. It allows teachers to type in pieces of text or poems they wish children to explore. The written material may be familiar or unfamiliar; it may be related to a project they are studying or simply may have an interesting language pattern or style. At first all the children see on the screen is the punctuation – commas, full stops etc. Even this stimulates discussion – is it a poem or a piece of prose? They make their decision and go to the 'scratchpad'– a type of notebook built into the program – to record their first predictions about the nature of the text. Now they must 'buy' letters. Every letter or letter pattern they choose to appear in the text costs them points. Every correct word they type in or correct guess they make on their note pad gains points. At first children tend to buy letters arguing about those which are 'the best value'. As they see single letters and groups of letters dotted about the screen, a pattern starts to emerge. The following letters at the beginning of a text:

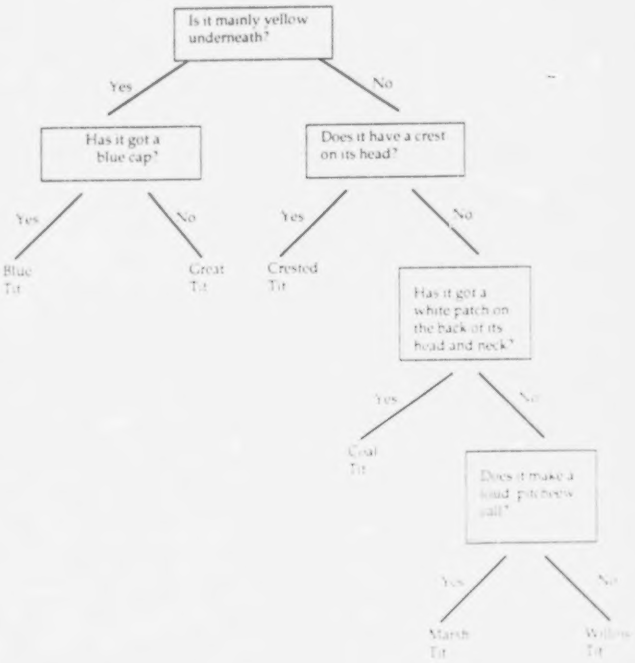
O–ce ––o– a –––e

may suggest the familiar opening to a traditional story – 'once upon a time'. The three letter word t-e may be guessed as 'the'. The children can type in any missing letters. Correct guesses are not only accepted by the computer but are also placed in the rest of the text. Thus one 'h' typed in a correct position places all the 'h's' in the passage. Incorrectly placed letters simply vanish from the screen. Thus the piece of writing is slowly revealed to the class like developing film in a photographers's darkroom (hence its name). There is great excitement as a word or phrase is identified or as the range of words suggests the general content being written about. Prawns, shells, fish for example might suggest a passage about the sea; it could however be about working on a trawler or in a fishmongers or part of a menu for a banquet. The children not only have fun watching the text develop before them but they also enjoy looking back at their notes on their scratchpad to see whether their guesses were right or wrong. It is not just an exercise in reading and comprehension; it is about collaborating and co-operating towards a common goal – and it is fascinating for teachers to watch all the skills and interactions generated.

As a supporting structure into which the teacher or indeed the children may place any text for exploration, this software is independent of country, culture or age range. It offers a stimulus to explore language from many different angles and from many different content areas. It can be used by groups of children just beginning to read or by groups studying an author's style or even to consider a mathematical argument. In this activity the computer plays the role of tasksetter and manager and provides support to a rich and enjoyable learning experience. It may be a task that an individual tackles, but equally small groups and large groups can combine their talents to find the hidden text. Teachers and children can work together if the text is not known to them. Thus there is the opportunity for the teacher to join in the activity as a fellow pupil rather than to share the role of a tasksetter with the computer. The children find that they can use the structure of language and their previous experience in language to help solve the problem. DEVELOPING TRAY stimulates communication and supports the strengthening of the use of structure in language.

2. Early science

The following description by Anthony Paddle describes work using EARLY SCIENCE. He considers a use of the computer that offers support to information structured in binary trees. A diagram such as this is shown below.


Is it mainly yellow underneath?
Yes
No
Has it got a blue cap?
Does it have a crest on its head?
Yes
No
Blue Tit
Great Tit
Yes
No
Crested Tit
Has it got a white patch on the back of its head and neck?
Yes
No
Coal Tit
Does it make a loud 'pitchew' call?
Yes
No
Marsh Tit
Willow Tit

Several years ago, various versions of a computer game based on binary trees called ANIMAL appeared in magazines and books. A typical dialogue with a computer running ANIMAL looks something like this:

```
Are you thinking of an animal?    YES
Does it live in water?            NO
Does it fly?                      NO
Does it walk on four legs?        NO
Does it go "BOING"?               YES
Is it a KANGAROO?                 YES

Are you thinking of an animal?
```

Only the boldface answers are typed by the player; the questions themselves are stored in the computer's memory. Things become rather interesting when you think of an animal the computer does not know about:

```
Are you thinking of an animal?    YES
Does it live in water?            NO
Does it fly?                      NO
Does it walk on four legs?        YES
Is it an elephant? (sic)          NO

The animal you were thinking
of was a?                         MOUSE

Please type in a question that
would distinguish an ELEPHANT
from a MOUSE?

      HAS IT GOT A TRUNK?

For a mouse the answer
would be?                         NO
```

In fact, the program starts each time knowing just one question and two animals. All the others are added by the players in the same way as the mouse. ANIMAL mimics a very simple learning process.

Clearly, the questions and animals are stored in the form of a key (or, equivalently, a binary tree), ANIMAL combines the functions of a key-searching program and key-building one. Although intended as a 'try to fool the computer' game, it could be used quite seriously as an identification aid. As such it has definite advantages over a traditional printed key, especially for children.

The first advantage is that only relevant questions are displayed on the screen. A key containing 1000 animals needs 999 questions but, in theory, only ten of them need to be answered to identify any one of the animals. In practice, reality triumphs over logic and keys cannot be designed that well; nevertheless, only a small fraction of the questions are relevant at one time. The remainder are distracting clutter, and it is easy to become hopelessly lost in a large printed key. ANIMAL avoids the problem by avoiding the clutter. Secondly, ANIMAL breaks down the highly abstract problem of designing an identification key into simple, concrete steps. To add the mouse to the key it is not necessary to think of all the attributes of mice or to search for some essence of mouseness that will distinguish mice from anything else. You are simply asked to find one clear difference between a mouse and one other animal. The key-searching part of the program ensures that the other animal is the most similar one already in the key, so that the mouse is inserted in the right place. This is not the only way of breaking down the key-building problem nor, if the aim is to produce 'elegant' finished keys, is it the best. Nonetheless it is easy and foolproof: If the individual questions work, the whole key will.

ANIMAL was not designed as a robust educational tool and suffers from a number of deficiencies. It is not possible to correct any of the questions or answers once they are entered – spelling mistakes are permanent. Nor is there a facility for saving a key on tape or disk, or for printing it out on paper. The language used by the program itself limits the use ('Are you thinking of an animal?' is a built-in question). It would be awkward to use it to classify plants or rocks.

There are now several elaborations of this idea, written for educational use, in which these problems have been solved. THINK is one example which, while keeping the outward key format of ANIMAL, has become a sophisticated tool for the creation, correction and searching of binary trees.

A further development is offered by SEEK, which comes in a package with THINK, several ready-made keys and a program called INTREE for typing in whole keys quickly. SEEK uses the computer's graphics to display the questions in binary tree form. The questions appear in boxes and, depending on whether you give a Y or N answer, you are led down a branch to the left or right into another box containing the next question or the answer. At any stage you can move back up the tree and down another branch, so that the whole tree can be explored. SEEK makes the structure of the information appear obvious. ANIMAL and its more direct descendants appear, by contrast to produce questions from nowhere; they seem cleverer than they are.

3. Using Key Handling Programs

Programs such as SEEK can be used in a surprising number of ways with children. Obviously they can be used to identify things if a suitable tree of questions is already available. At the other extreme, children can build their own tree from scratch, given a set of rocks, twigs or kitchen powders, for instance.

There are also strategies that fall between these two. If a class is planning to go pond-dipping, a key to the commonest animals might be created in advance, using information from books. New animals can be added one at a time as they are found, possibly over a long period. This may well be the best approach with large, complex groups of things. The initial skeleton tree can be designed so that its main branches represent the major groups (nymphs, larvae, snails, worms, etc.) and the research involved in creating it can give focus to the childrens' preparations for the first outing.

In the classroom, identification exercises can provide very effective frameworks for practice of observational and experimental skills. A particularly good example is the POWDER tree supplied with the SEEK/THINK package. On the surface it is simply an identification key for common household powders, such as sugar, salt, washing powder, flour and baking powder. The questions, though, are not just passive observational ones: Most of them ask the children to do something to the powder and watch its reaction. In the next column is part of the key as produced by SEEK on a printer.

	QUESTION	YES	NO
1	Feel your powder? Is it smooth or floury?	2	3
2	Put some in a teaspoon and heat over a candle. Can you see lots of steam?	BAKING POWDER	4
3	Look through a magnifying glass to see if it is lumps or crystals. Is it crystals?	5	6
4	Put a drop of iodine on your powder. Does it go blue/black?	FLOUR	ICING SUGAR
5	Put some in a teaspoon and heat over a candle. Does it smell like toffee?	SUGAR	7
6	Put some in water and shake. Do you get lots of bubbles?	SOAP	POLY-CELL
7	Put a teaspoon of powder on a saucer and add vinegar. Do you get bubbles?	WASH-ING SODA	SALT

There is no one way of classifying things. There may be generally accepted ways for groups like plants, animals or rocks, but even these are subject to constant argument among scientists. If children are to understand why things are classified the way they are, they need to explore and compare different ways. It is here that programs like SEEK display their real value. By taking care of the overall organisation of the tree, they let the children concentrate on close observation, comparison and the logical and language aspects of choosing good questions.

Imagine that a group of children are trying to identify some epsom salts using the POWDER tree. They will probably find that it is wrongly identified as a salt. If they decide to extend the tree they will be asked to find a question to distinguish the two. This is no small challenge, finding the best question may take a lot of time, experimenting and discussion. The first stage is to find out everything they can about the two substances by observing, practical testing and research into their uses. The result may be quite a long list of differences, so the second stage is to decide on the best question to be added to the tree.

'Does it dissolve in water?' is no good because the both do.

'Does it taste salty?' may be ruled out on safety grounds (someone may try to identify something poisonous).

'Do you buy it at the chemist?' requires prior knowledge and would be impossible to answer if you really did not know what the powder was.

'Does it have big crystals?' does not have a clear answer: It depends what you compare them with. Also the crystal sizes of both vary enormously.

'Does it have long, thin crystals?' is better, as is 'Does it turn into white cake when you heat it over a candle?'

Some of these problems are quite subtle, and children are unlikely to spot them until they try the 'bad' questions in a complete tree. Fortunately, all the more recent programs let you prune and repair a tree without having to rewrite the whole thing; so children can learn from their mistakes and correct them with a minimum of frustration. A good way of identifying problems and sharing insights is to encourage groups of children to test each other's trees.

4. Mathematics

AUTOCALC is another example of a simple program that promotes considerable discussion and sharing of processes. It enables children to articulate their own methods and ideas and has proved an extremely valuable way to build their confidence in their mathematical abilities. The children are challenged by the program to try out their mental arithmetic skills and to review and compare the range of possible processes. A large screen is needed at the front of the classroom. The screen presents the problems in the following format:

$$\begin{array}{rr} & 44 \\ + & \underline{29} \\ & -\,- \end{array}$$

After a delay the computer then supplies the answer to the calculation

$$\begin{array}{rr} & 44 \\ + & \underline{29} \\ & 73 \end{array}$$

The mode of the program is to generate such problems by selecting random numbers according to the parameters set at the beginning, using a chosen operation and displaying the answers after a chosen time delay. The option screen used for defining the type of problem to be set is shown below:

Autocalc Options

Type of problem	Subtraction		
Difficulty Level	Own		
Top number	1	to	20
Middle number	1	to	10
Bottom number	0	to	20
Delay time	2 seconds		

This option setting provides simple subtraction problems for young students.

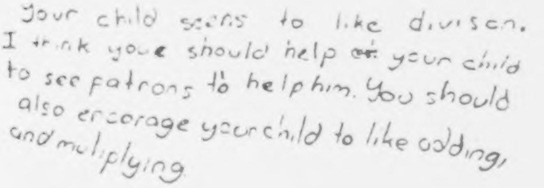
Imagine a class of children working on the ways in which they 'add 9' to numbers. The computer is set to produce problems where the number is generated between 0 and 99, the second number is fixed at 9 and the time delay of 3 seconds before the answer is given has been set. Fifteen problems appear one after the other and the children attempt to calculate the answer before it is displayed by the computer. To simulate the experience complete the following problems as quickly as you can:-

$$\begin{array}{cccccccc} 28 & 90 & 32 & 77 & 88 & 79 & 37 & 66 \\ \underline{9}+ & \underline{9}+ & \underline{9}+ & \underline{9}+ & \underline{9}+ & \underline{9}+ & \underline{9}+ & \underline{9}+ \\ \\ \_\_ & \_\_ & \_\_ & \_\_ & \_\_ & \_\_ & \_\_ & \_\_ \end{array}$$

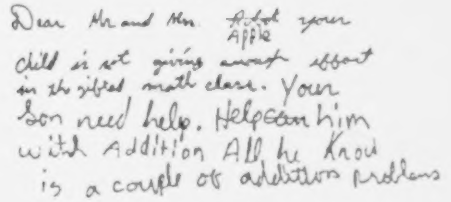
Probably after the first try at this task the children will feel that they might be able to get the answers in under 2 seconds so that they can have another go with reduced time delay. Some might even like to go at producing an answer in 1 second! After this activity the children are asked to say exactly how they got the answers. The following list of methods was the result of a class of ten year olds sharing their ideas:

1. Helen decided to add one to the 'tens' and then take one away from the 'units'.
2. Jonathan was happy to count on his fingers but didn't always have time.
3. Susan added 3 three times.
4. Jo subtracted '1' and added '10'.
5. Anne worked out 'how many to the next 10's'. This is then subtracted from 9 and the remainder added
   e.g. $78 + 9 = 80 + 7 = 87$
6. Simon added 2 four times, then 1.
7. Jane used different mathematics for different problems.
8. Michael just 'knew' the answers!

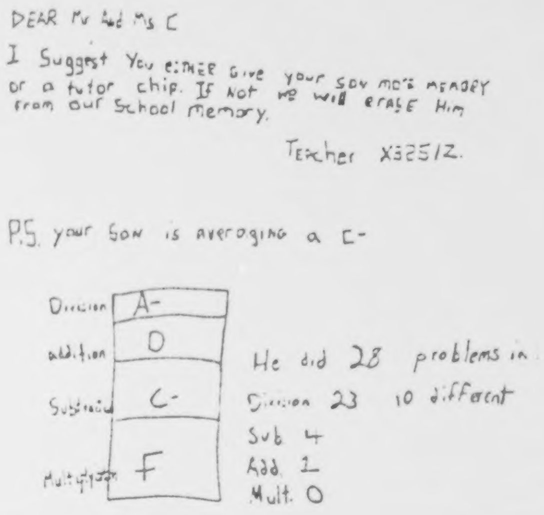
The children greatly enjoy sharing their methods and trying out each other's ideas. They are also encouraged to use calculators – various tasks are given that begin to expand their understanding of number and to encourage them to feel confident enough to move into estimating outcoming numbers as well. AUTOCALC can be run in a mode where the challenge is not to do the calculation but to spot every time an incorrect answer is displayed. (This is called the 'oops' mode.) The skills needed to succeed here are now dependent on having a good grasp of number bonds and relationships. Another excellent activity is to ask the children how many problems they can make combining a number between 0-99 and a number between 0-9 that has the answer of 5; e.g. 13 − 8 = 5; 1 × 5 = 5 etc. After exploring the problem in groups, the children offer their solution. This brings out some fundamental mathematical processes – classifying sets within the solution set – setting the initial conditions in order to limit the solution set to a finite set and many others. The children express these ideas in their own language and, of course, they are not yet aware of the generalisation of such ideas. However it is at this point that we become aware that this simple computer program has given the children a stimulus that has caused them to become true mathematicians. In sharing their mathematical processes and in valuing each other's ideas they will build up confidence in their own abilities to offer something to the subject. In this way we can begin to remove the fear that so many people leave school with in regard to their mathematical abilities. A final stage to the discussion of the '5's' problem is to watch the computer doing the same problem and to write report to 'its parents' on its performance. As the computer applies an extremely simple algorithm (it just keeps randomly generating problems but only displaying those that give the result of 5), its performance is certainly open to criticism. Here are some of the children's reports:



Your child seems to like divison. I think youe should help ~~on~~ your child to see patrons to help him. You should also encorage your child to like adding, and muliplying



Dear Mr and Mrs ~~Robot~~ Apple your child is not giving enough effort in the gifted math class. Your Son need help. Help can him with Addition All he Know is a couple of addition problems



DEAR Mr And Ms C

I Suggest You EITHER Give your Son more MEMORY or a tutor chip. If Not we will ERASE Him from our School memory.

Teacher X32512.

P.S. your Son is averaging a C-

Division A-
addition D
Subtraction C-
Multiplication F

He did 28 problems in
Division 23 10 different
Sub 4
Add 1
Mult. 0

Critics might say that the activities promoted with AUTOCALC are not valuable because they are dealing with numbers out of context to any real problem. However we hope that the examples here, which are only a minute part of the range of possibilities, show children becoming aware of their own power and thought processes and also taking over a range of 'teacher roles' at various stages. Feedback to the teacher of the children's reasoning and the way in which they articulate this is a major contribution of AUTOCALC.

A few years ago Michael Girling (Her Majesty's Inspector) suggested that a definition for numeracy might be 'appropriate use of an electronic calculator'. What number sense would one need in order to qualify?

We suggest:

1. Instant command of single digit arithmetic
2. Command of basic multiplication facts
3. Skill in estimation
4. Capacity to spot errors
5. Capacity to select which operations are appropriate in any problem

With the exception of 5 all these points are strengthened by the activities possible with AUTOCALC.

Concluding Remarks

This section has taken just a few examples of simple software to illustrate how computers can have a stimulating and refreshing relationship with children. We are keen that the computer becomes an accepted assistant and friend of both teachers and children. The use of Logo, data banks and word processing, have not been discussed here as many books and articles are available to the reader on these topics. Such languages and systems can be employed to stimulate discussion and exitement such as is described here. However, they can make considerable demands on the users and we would recommend that subsets of such systems are used to start with. Slow progress is being made with implementing a curriculum that make effective use of computers and calculators. This is due to the fact that there is not as yet a great deal of curriculum support materials to introduce the range of learning activities that simple or complex computer software can support. However, this material will gradually emerge and there is certainly enough available to any enterprising school to offer children the advantages of a computer in their classroom.

Any school able to equip each classroom with a single microcomputer would gain experience and confidence within a matter of months rather than years. Add to this provision a small laboratory for word processing etc. together with a collaborative staff exploring possibilities together and the scene will be set for an exiting time for children in such a school.

Secondary School

Phenomena, Theories, Experimental Mathematics

In mathematical knowledge one can differentiate between facts on the one hand and the insight into their necessity and their connections on the other hand, or between phenomena and theories. This distinction becomes clear, for example, in the domain of the geometry of triangles: Examples of phenomena are the observable facts, such as that the three angel bisectors meet in one point and similarly for the perpendicular bisectors, that the sum of the inner angles equals 180 degrees, that two triangles which have the two sides and the enclosed angle equal have all other measurable parts equal, the formula of Pythagoras, etc. Most classical theorems of school geometry belong here, but so also do more qualitative facts such as: If two sides are fixed in length, then the third side gets longer if the enclosed angle is made bigger (up to 180 degrees). There is now special software such as The Geometric Supposer or Cabri Géomètre which helps to find such facts by giving assistance in the making and systematic variation of geometrical constructions.

In the domain of theory there is the logical ordering of facts (local and global), the insight into the necessity of observed facts, the determination of the proper conditions under which the facts remain true (the domain of validity), etc. As a concrete example, let us look at the calculus (analysis). Phenomena are: The graphs of functions, say of $f(x) = x \sin 1/x$, the fact that $\sin x/x$ tends to 1 as x tends to zero, the divisibility of $x^n - 1$ by $x - 1$ and the form of the divisors, the formulas for the derivatives of elementary functions, the linearity of the integral, or the shape of solutions to a specific initial value problem for a differential equation.

To the domain of theories, there belongs the definition and fundamental properties of the limit, the completeness of the real numbers, the definition of the integral, the limits of validity of theorems, and explanations of facts by arguments.

It is interesting, that there may be different possible theories, for example, Euclidean or Cartesian geometry, with formalist or constructivist foundations. Or, in the case of analysis there are different possible non-equivalent theories, the classical $\epsilon - \delta$-theory, non- standard analysis and different constructivist approaches. But all those different theories explain – in different ways – the same phenomena. And all the concrete applications of geometry or calculus only rely on the phenomena, not on the underlying theories. In a similar way, computers and mathematical software work exclusively in the realm of the phenomena; they can only exhibit phenomena. And they are able to show the phenomena even to students who have not yet mastered the theory.

This is the point in our argument: In a mathematics class using mathematical software, students will get to see and know a lot of mathematical phenomena. The mathematical theory then has to explain these phenomena; thus mathematics shifts in the direction of a science which orders, describes and makes understandable facts that are already known and obvious even without explanation. This is in sharp contrast to classical teaching methodology, especially in such domains where it was hard to approach the phenomena without theory or advanced technology.

Here is an example. In the study of functions and their transformations, traditional teaching deduced behaviour mostly from theory, since the actual plotting of function graphs by hand was far too expensive, in terms of time and labour, in order to make students see the facts, for example, the graphical translations connected with the transformation $f(x) \to f(x + a)$. With the help of a function plotter they may observe those transformations, first connected with a concrete f and a, then systematically explored with free chosen examples, and in between also formulated as hypothesis and verified by arguments. In this way, the temporal order of phenomena and theories reverses, and gets closer to the usual habits of mathematics as a research activity. Of course, such an approach has often been used with mathematical content where exploration of phenomena was cheaper.

The didactical paradigm just described has often been referred to as "experimental mathematics", but it has to be stressed that theory is an indispensable part of it in order to be mathematics. Just playing around with a function plotter does not necessarily lead to insight. You normally need hints, ideas, hypotheses, questions in order to see something and get involved. (See Goldenberg, 1988 for more specific considerations and examples.) As a counterexample, using fractal generating software may give spectacular pictures of great esthetic value but, if you stop at the phenomena, you won't get at mathematics with such software. You need at least general concepts such as self-similarity or symmetry, which are also needed for the better understanding and appreciation of the beauty of the pictures involved.

Software for Secondary School Mathematics

We shall discuss this for three content areas – Geometry, Functions and Data Analysis.

Geometry

Two software packages for geometry education were mentioned above: The Geometric Supposer and Cabri Géométre which allow constructions of most of the problems of Euclidean plane geometry. A so-called draft mode allows the exploration of consequences of moving one point in a figure while keeping its connections to other points (see Fig. 1). Descriptions and examples are given in Schumann [1990]. Here we shall describe two other pieces of "teachware", which allow some unconventional activities which are closely related to the curriculum for grades 7 and 8.

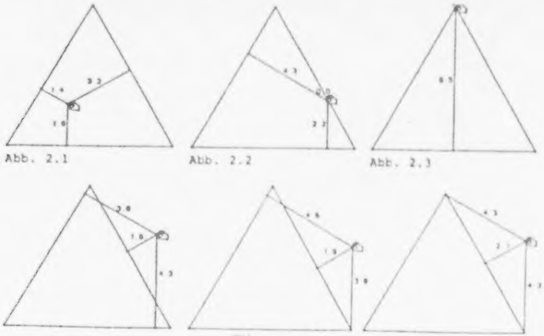


Figure 1

The elementary didactical philosophy is that there should be two levels of action in geometry classes, when using a computer: On one level the pupils should learn the constructions manually with ruler and compass, as usual. On another level they improve their competence with these constructions by solving geometrical and applied problems with graphics procedures on the screen which they perceive as efficient and comfortable tools. In particular, this use of computer graphics in the early years of secondary school has proved useful in three modes:

1. Using procedures for ruler-and-compass constructions which have already been understood as building blocks for more complex constructions without the need to repeat the elementary constructions again and again.
2. Using procedures for constructions in ways which cannot be realized with ruler and compass.
3. Using procedures for large and technically difficult constructions, which demand many iterations of elementary constructions.

The Geometric Supposer fits in mode 1. We now discuss two other software packages, SYMMETRIC TURTLES and KALEIDOSCOPE, which illustrate modes 2 and 3.

SYMMETRIC TURTLES (Graf, 1988)

It is well known that Logo's turtle graphics can help at the beginning of geometry education. As a tool which provides an extension of the ruler and compass a "running turtle" has been developed. This follows the concept of Abelson's dynaturtle [Abelson and di Sessa, 1985], but without inertia. To some extent you can use it like a pencil, controlled with keys.

Keys 1, 2 .. 9 put it in slow or faster forward motion on a straight line, key 0 stops it. Z or N lets the turtle draw or not draw when moving. A, S, D, F effect small (5 degrees) or larger (15 degrees) left or right turns of the stopped or moving turtle. Q marks the position of the turtle on the screen and determines a number for this point. This point can be reached again via keys K or P. K turns the turtle in its actual position heading for another point. This corresponds to putting a ruler through two points. P puts the turtle on an already marked and named point. And so on.

This running turtle allows construction of many figures of interest in plane geometry. Besides this turtle there are two more turtles, an "axial symmetric turtle" and a "central symmetry turtle". They are controlled in the same way as the standard running turtle. But they do not only draw the figure controlled but also the figure's symmetric image in a different colour relating to the x- or y-axis or the centre point of the screen. This happens simultaneously and pointwise. This mode of construction for symmetries helps the user to recognise the properties of the mappings immediately and more easily than from the final picture. You see that a straight line remains a straight line, you see how the direction changes under axial symmetry and how it remains the same under central symmetry. You also see that a straight line and its picture are parallel under central symmetry, but have different directions, etc. Figure 2 contains some examples. Unfortunately, the "dynamic" quality of the turtles cannot be seen from these figures.

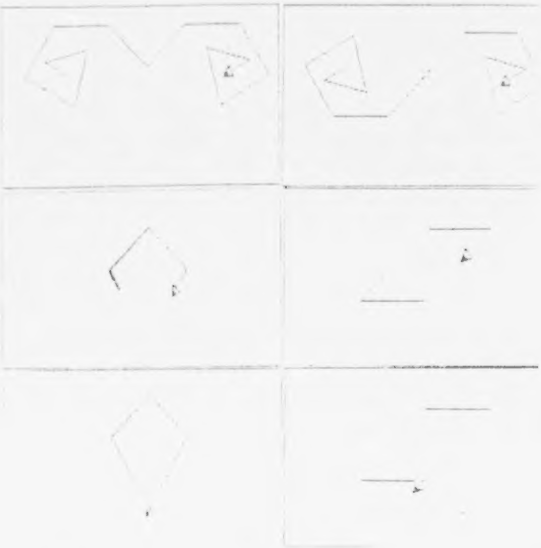


Figure 2

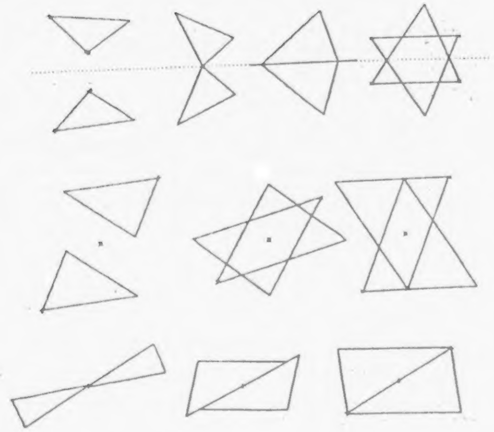


Figure 3

Figure 3 shows how the following question can be examined: "What happens when reflecting a triangle in different positions relative to the axis of symmetry or a point?"

Figure 4 gives a systematic answer to the question, "How can quadrilaterals be generated by reflecting triangles?"

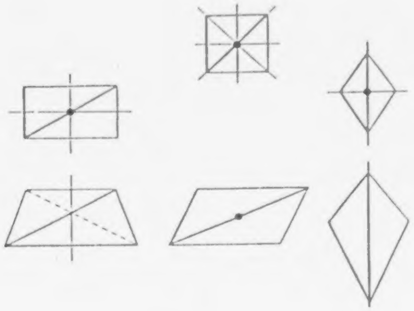


Figure 4

First, it is convenient to choose a side of a triangle as an axis of symmetry. Then with the turtle you get a kite. The special case of an isosceles triangle occurs if the angle adjacent to the axis is 90 degrees. If this angle is greater than 90 degrees, then you get a quadrilateral which is not convex. You can also get a rhombus and square by starting with special triangles. But you never get a general rectangle or a parallelogram or a trapezoid. The central symmetry turtle, however, applied on the centre of a side of a triangle produces a parallelogram immediately. This is an exciting discovery. The choice of this special point of reflection is suggested by the experiments shown in Figure 3. Again, no trapezoid occurs. This fact can result in geometrical discussions. More details about these tools are given in Graf [1988]. Some reactions of teachers and student teachers to this kind of teachware and some experiences in classes are also reported there.

KALEIDOSCOPE

In a paper by Graf and Hodgson [1990] it is shown that the kaleidoscope can be a window to some geometric concepts. These are elementary and rich at the same time. They are rich in the sense that they offer not only a mathematical model of the kaleidoscope but also models for other worlds like planes to be tiled or even like a fictional kaleidoscope.

From a methodological point of view the mathematical problems connected with kaleidoscopes can be worked on at the following five levels:

1. Looking through the real kaleidoscope.
2. Reducing the kaleidoscope to a model with two or more real mirrors placed on a sheet of paper containing some figure.
3. Abstracting the mirrors and their reflections on straight lines (axial reflections) constructed with ruler and compass.
4. Transferring these constructions to a computer graphics display.
5. Using formal methods to describe the phenomena (and PROVE theorems!!!), for example, those of analytic geometry and linear algebra.

Here we can only give a few glimpses on the software for simulating kaleidoscopic phenomena on the computer and examples of patterns that can thus be produced.

a) Two-mirror kaleidoscopes: The main menu offers a choice of four different types of kaleidoscope. Mode 1 leads into a dialogue about forming a kaleidoscope with an arbitrary angle. The user gives the positions of the axes and then the position of the object to be reflected. The computer then displays the two axes and the object. It then constructs and displays one reflection after the other until the pattern is complete. This can be done with a pause after each image or in an automatic mode. Mode 2 allows one to select a kaleidoscope with angle 45, 60, 72, 90 or 120 degrees and then proceeds as above. Figure 5 shows some steps in the development of a 60 degree pattern and the same process for a 70 degree pattern. One of the many questions which will arise after such experiments is: How many reflections are there before the pattern begins to repeat itself? Circular symmetry can be discovered and discussed after such experiments.

b) Polycentral kaleidoscopes: These are built from a greater number of mirrors and thus produce groups of images around several centres spreading in all directions. Forgetting about special objects between the mirrors and just regarding the reflections of the triangles or quadrilaterals and so on shaped by the mirrors you discover another mathematical phenomenon: Coverings of the plane. In the case of three equal mirrors or "sides" you end up with a perfect tiling of the plane. Figure 6 shows the growing of such a tiling. Obviously a discussion will arise from this about good and bad tiles.

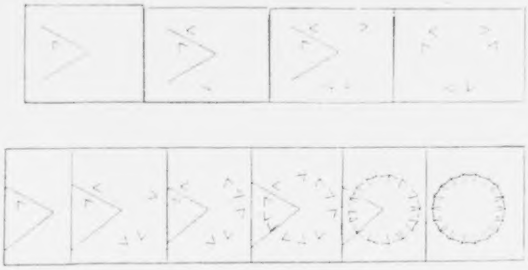


Figure 5

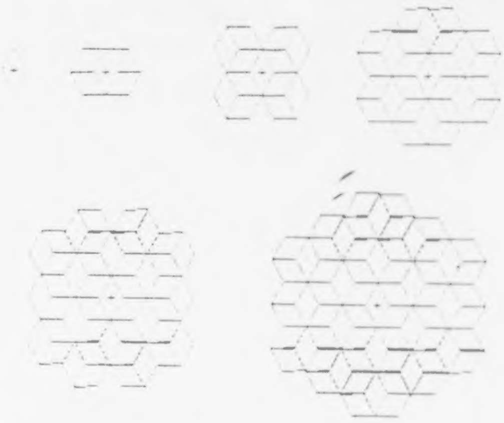


Figure 6

c) Fictional kaleidoscopes: The examples given so far considered the transfer from real kaleidoscopes to mathematical models, combining axial reflections and varying the type of kaleidoscope. Why not vary the mathematical model and forget about reality? Central reflections (half-turns) will give us a model of some fictional kaleidoscopes having no physical counterparts. One case is to look at a triangle again, determined by three centers of reflection, and see what happens after repeated reflections. We can get a pattern extending throughout the plane (see Figure 7), leaving some blank spaces.

A new situation occurs if we start reflecting a triangle not about its corners but about the midpoints of its sides, and go on reflecting the images about these midpoints. A new tiling of the plane develops and – what is really surprising for the beginner – this works well with any triangle. Next you can turn to quadrilaterals and again you discover immediately that a perfect tiling can be completed with any quadrilateral, even a non-convex one (see Figures 8 and 9). So the fictional kaleidoscope brings you back to a real problem and the search for its correct mathematical solution.

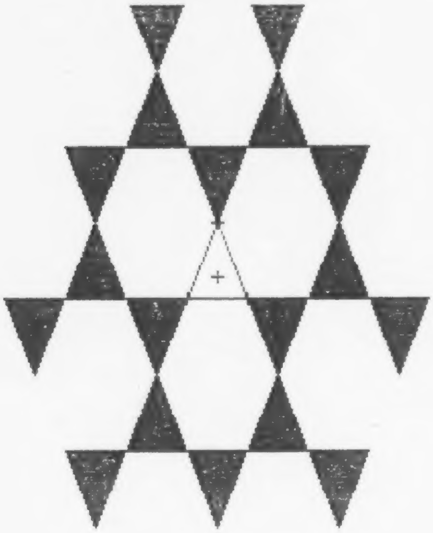


Figure 7

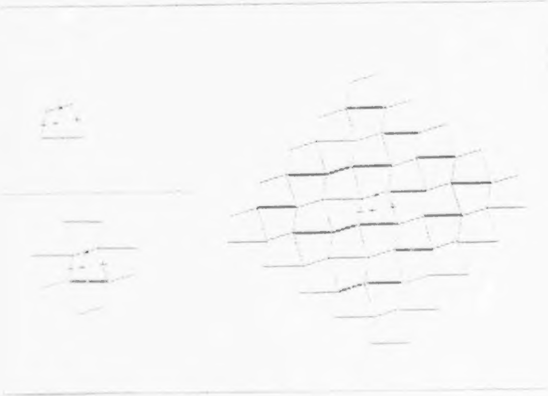


Figure 8

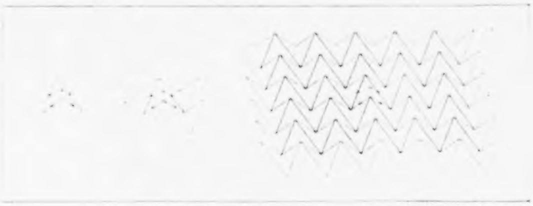


Figure 9

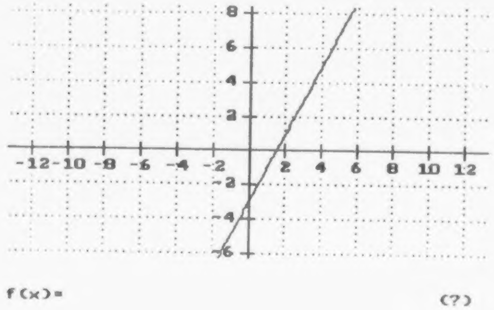
We conclude this section about geometry with some general remarks. There should always be a very careful examination of the advantages for learning before the computer is used in some field of mathematical education. There is no use in transferring manual or mental activities (like constructions with ruler and compass) to the computer unless this brings about more efficiency in learning. Another good reason for using the computer may exist if the computer allows activities which the students cannot achieve with their hands or brains. Then the computer acts like an additional tool, increasing the traditional abilities of the students. SYMMETRIC TURTLES and KALEIDOSCOPE are good examples of such tools. They allow

- additional help in exploring mathematical problems.
- a great variety of investigations with little effort
- easy experimentation
- the viewing of a very broad spectrum of complex geometrical constructions which turn up when studying reflections of complicated figures
- doing manually impossible constructions like the pointwise simultaneous construction of two or more figures
- introducing and using simple methods of CAD (computer-assisted design), a technique which has replaced manual technical drawing to a considerable extent.

Functions

Function plotting software in acceptable quality for use in schools is now available for nearly all modern microcomputers, and there are now several sources of didactical material describing teaching units, giving hints and providing exercises which can be used by normal mathematics teachers. The general idea of a function plotter, to plot the graph corresponding to a user given function, can also be inverted, namely to plot the graph and let the user look for the term. This is realized in the "Funktionen raten" (looking for the formula) part in *Graphix*.

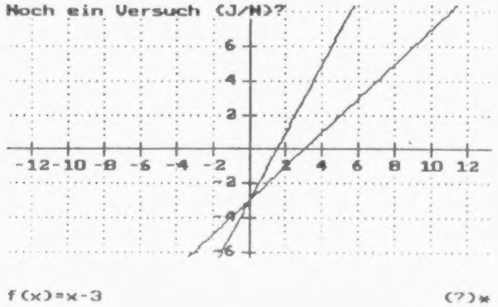
For example, the program plots the graph of a function – say $f(x) = 2x - 3$ – but does not show the term (Fig. 10). The user has to make use of the information given in the graph to guess the function term and put it in. The computer reacts by plotting (in another color, if available) the graph corresponding to the user's term in the same coordinate system. If the user has not got the correct solution (Fig. 11), he or she can now see the difference between the original and the guessed graph and use this information to debug, that is, to correct any mistake. As many tries as desired can be made. It is also possible to wipe out the screen and see only the original function.



f(x)=

(?)

Figure 10



Noch ein Versuch (J/N)?

f(x)=x-3

(?)*

Figure 11

The functions plotted by the program are offered in different sets, organized according to difficulty and type of function (linear, quadratic, cubic, trigonometric, exponential, using absolute values, etc.). The sets can be changed or augmented with a simple text editor by the teacher, according to the needs of the students. A specific option lets the program be used by two partners (individuals, groups): the first gives the term and the other has to guess it from its graph.

The simple idea of the program gains its motivational and challenging character from the use of a sophisticated function plotter, which comes close to the accustomed appearance of terms and graphs, and from its deliberate generosity to an inexperienced user. It is simple to use. The user is not penalized for wrong answers. And it has adequate error control, not through comparing the user's term with a predefined list of possible right terms, but by numerically comparing the graphs with a certain tolerance. So the software aims really to help users to evolve and debug their knowledge about elementary functions and their standard transformations.

The program is to be used mainly by individuals or small groups, in a wide variety of levels, grades 7 to 12 and up. It may be used for drill and practice, and, of course, for remedial work.

Data Analysis

Statistical education – as mathematics education in general – often has to cope with the problem that, in order to solve real problems, the necessary techniques are taught and, in consequence, also understood by students in isolation; their proper conditions of application, their region of validity, their limits are perhaps theoretically known, but seldom part of active knowledge. In order to overcome such limited understanding, one method is to confront students with problems connected to themselves, so that they don't take the methods as neutral, but of real importance. One of the goals of the software *Times* is just to give students some real data, connected to themselves, in order to analyze and to draw conclusions from the data and thereby about themselves.

The software allows experiments with reaction times: The computer produces a specific signal and one of the students has to react in a specific way, for example, by pressing a specified button, and the computer measures the reaction time. The process repeats, and the data are stored into a file bearing the name of the student. Another student does the same procedure, and the data is compared. Which student is better? Is the arithmetical average a fair arbiter or is the median better? How should one judge extreme values? The program offers several methods of comparing data, including some well known statistical techniques. It calculates diverse quantities such as averages, variances, the plot of one distribution of values against the other It does QQ-plots, displays the data as time series, etc.[1] In defending their results, the students hopefully learn to judge cautiously, to see the techniques as helpful but normally not decisive tools, and the necessity of properly interpreting the data rather than automatically drawing conclusions after a routinely applied test.

General Tools and Methods

Besides studying software for specific mathematical areas like the ones just discussed, it is important to consider software which supports specific mathematical methods which have importance in different areas. Here algorithms in their original sense (think of Euclid's algorithm) are most familiar and were integrated into mathematics education even before the advent of computer systems. First we shall give an example of an algorithmic strand, which fits the curriculum for the German "Gymnasium". From this you will be able to see how this old mathematical idea of algorithm can be extended to a number of complex mathematical problems. Then we shall discuss the general problem of how to combine in class teaching students to understand and execute mathematical methods and to solve mathematical problems (from multiplication to integration) by hand or brain with the need to tell them that there are computers which can do these things easily if you just give the problem to them in the proper way. This is the black box/white box problem. Finally, we discuss two more general methods of growing importance in mathematics education (as well as in mathematics research) - simulation and model building.

The algorithmic strand.

Algorithms are patterns with a certain schematic background; although high mathematical invention was necessary for their discovery, only stupid and exact processing is needed for their application. With this didactic philosophy the teaching of concepts and theories of mathematics had priority at schools. The use of algorithms formed the center of exercises, homework and control of achievement and so pupils were educated as if they were little computers. Related to this secondary role of algorithms is the fact that several thousand years of history of mathematics have not produced a uniform language for the description of algorithms. Now there is a continuous algorithmic strand which forms 15% of the curriculum in mathematics education during the nine years of German grammar school. (We begin at year five since we do not consider the four years of primary education.) The following list shows typical algorithms and also their related subjects.

5 Relations between the fundamental arithmetical operations
Transformation between numbers with different bases: (10,2,5,16 etc.)
Division algorithm
Sieve of Eratosthenes
Optimizing terms
Summation of arithmetical series according to Gauss
Fundamental operations with sets

6 Calculation with fractions (handling formal rules)
Greatest common divisor and least common multiple (algorithm of Euclid in several variations)
Prime numbers, twins of prime numbers, distribution of prime numbers etc., factorisation of numbers
Arithmetic means, relative frequencies
Diagrams of descriptive statistics

7 Tables of proportions
Calculation of percents and interest
Random experiments
Constructive geometry in two dimensions
Geometrical mapping

8 Algorithm of Heron
Iterations for linear equations
Symbolic processing with equations

9 Solution of quadratic equations
Graphs of quadratic functions
Combinatorics
Continuation of geometry (similarity)

10 Several methods of integration of the circle
Division of polynomials
Trigonometric construction
Descriptive statistics

11 Experiments with sequences and series
Discussions of functions
Algorithm of Newton with variations
Regula falsi
Methods of optimisation

---

[1] For a more detailed analysis and critical description see Biehler and Winkelmann (1988).

12 Methods of integration according to Simpson, Gauss, Romberg etc.
Symbolic integration
Algorithm of Gauss for systems of linear equations with variations
Operations with matrices etc.

13 Stochastic simulations
Symbolic handling of limits (l'Hospital)
Standard methods of inductive statistics
Methods for numerical, graphical and symbolic solution of simple ordinary differential equations

This listing refers to a basic level of higher education. For the advanced level ("Leistungskurs" with 6 lessons in the week) a lot of possibilities can be added in the last three years such as:

> complex numbers, special numerical methods, algorithms of the theory of graphs, fitting of curves according to Taylor and Gauss, interpolation of functions according to Lagrange and Newton, cubic splines, study of nonlinear iterations, mapping and representation in three dimensions, constructive non-euclidean geometry etc.

Also in the content of an algorithmic strand, the methodological aspects need not be lost. Several basic formulations of computer algorithms are helpful for mathematical comprehension, too. For example, pupils always have difficulties understanding the usual notations of sums, double sums, etc. The algorithmic notation using for-loops or nested loops removes many difficulties in understanding the role of summation index etc. The practice of programming recursions is helpful for understanding the logical basis of induction proofs, etc.

For nearly all of the subjects listed software is available, some of it with more options than are needed in schools. Most teachers are pleased that they do not have to enter into the specialities of graphic representation and the other "higher" work of computer insiders. Still some of them remember another kind of work only a few years ago: For important algorithms of mathematics (Euclid, Gauss, Newton, Simpson etc.), teachers themselves had to write their own programs. The advantage of this was that they could develop the central ideas simultaneously in their classes and in the programs. The disadvantage was that the handling of many programs was not easy. Still, a further advantage was that the teacher could modify an algorithm using the (sometimes unusual) suggestions of the pupils. As an example, in Newton's method for the solution of transcendental equations, you could take the tangent of a function instead of the function itself. Or you could take tangents with the same constant slope as the first tangent (the method converges in many cases). Alterations of this kind are generally impossible with acquired programs, which seldom allow such open didactical processing. Naturally, for these purposes the teacher needs a simple and transparent computer language with natural keywords and sufficient mathematical operators as well as a compiler which can understand the language in the same sense as humans. Teachers need as well a good cooperation with teachers and pupils of computer science, who can construct good programs according to their desires. Some programs in the school market need to have a didactical dimension so that, for example, the plotting of functions can be stopped and continued using the intuition of the pupils. During algorthmic processing intermediate suppositions about the results should be possible which can be verified or falsified.

Symbolic Processing/Symbolic Manipulation

In recent years symbolic processing for personal computers has entered into schools (see the chapter by Hodgson and Muller). Solving linear and quadratic equations, equations of third and fourth degree, large systems of linear equations, simplification of rational expressions with "towers" of double fractions, division and simplification of polynomials can all be done with symbolic algebra, often integrated with the direct processing of very large integers. Where exact methods fail, approximations are possible. Symbolic differentiation and integration, symbolic vector analysis and, finally, the symbolic solution of ordinary differential equations of first and second order together allow the possibility of ignoring all the rules of school mathematics in a traditional sense. These packages are made by professionals. Therefore, they often do not present intermediate steps and some other didactical remain. Some of the symbolic packages are not programmable by the user. Nevertheless the union of numerical, graphical and symbolical tools has enormous power for schools.

Enlightenment through Black Boxes

In a recent article, Buchberger (1990) asks, "Should students learn integration rules?", given that now there are computer algebra software systems available which solve any integration problem much more quickly and more reliably than any student could ever do with paper and pencil. Buchberger immediately generalizes the question for all those areas of mathematics which are "trivialized" by modern software, especially computer algebra systems. He answers – for mathematics and computer science majors – with his "White Box / Black Box - Principle": Students should learn the theories and algorithms of such an area first, using the software only for subordinate tasks (e.g. partial fraction decomposition) but, after having studied the area, all calculations from this area should be left to the computer.

For schools and general mathematical software the situation is more complicated: Numerical and graphical oriented software doesn't trivialize an area of mathematics, but may provide profound help in solving problems; school mathematics does not only provide mathematical theories and algorithms but also their intended applications, their modes of use and the translation schemes needed in using them outside of pure mathematics. High school students are just not future mathematicians, but could be regarded as future users of mathematics as well, who obviously should have a different attitude towards mathematical tools. So here we do not give a recipe but rather some considerations which might help in coping at school level with the problem of using ready made software which cannot be made "translucent", since the details may be too complex, or totally hidden from the user, or just not worthwhile studying for secondary school students.

First of all, using ready-made mathematics, even if not fully understood, is to be seen as taking part in mathematics as a social enterprise. It may be looked on as part of teamwork: Users rely on professional mathematicians and programmers. But the cooperation is anonymous since the user can't talk to "coworkers", and users have to know a lot in order to use the black box correctly and with beneficial results. But knowledge about black boxes (procedures, algorithms, etc.) can be of various kinds:

Logical or external. The user knows the mathematical specification of the result the software delivers, but doesn't know the method by which it has been achieved. This is the classical black box and is usually the case with the use of computer algebra systems or simple calculators. A symbolic integration can be understood (and independently checked) even if the internal Risch algorithm isn't understood or its existence even known. The cosine of a number can be interpreted correctly as the best approximation within the domain of machine numbers to the correct real number, etc.

Analogous. If a complete specification of the result of the software is not available, an analogous knowledge of a similar algorithm may often help. The graph of a function, as displayed by a function plotter, is different from the graph of the function as normally defined within mathematics. But the experience of doing function plotting and a reflection on the possible pitfalls (e.g. vertical asymptotes, discontinuities or the proper determination of maxima) may help in understanding results and becoming aware of possible limitations. For the normal student it is not worthwhile to learn the special tricks and algorithms programmers of function plotters use to give reasonable results even in difficult situations. Analogous knowledge is needed in general in the use of numerical software – possible pitfalls, trade-offs between step widths and obtainable accuracies, between reliability and speed, etc.

Algorithmic. Here the user knows – on a certain level – the specific algorithmic approach used by the software, for example, that the numerical integration software uses Simpson's rule, which the use had applied in some hand calculations. But for a suitable use of the software, the user has to have some more general knowledge, too – the approximation character and the order of the algorithm, its domain of validity, in what circumstances to switch to other algorithms, etc.

All three kinds of knowledge have their special value, and in most circumstances they should complement each other. There is no *a priori* best way of enlightening a software black box. Of course, mathematics teaching has the duty to enlighten black boxes, to make them grey at least, but in which way and to what extent has to be decided in view of the intended use of the software, the kind of knowledge to which this new knowledge is to be added and connected, and to the overall goals of mathematics teaching in the specific age group and school system in particular.

On the Concept and Importance of Simulations

How does one simulate a dynamical process? Such a process is described by specifying the transition from one state of the system to the "next" state; mathematically this is done by (systems of) difference or differential equations. In order to simulate such a process, one first has to specify all parameters, initial states and possible external influences numerically, and then follow the evolution of the states numerically, replacing all mathematical operations which have no direct arithmetical translation by numerical approximations. Some ending conditions have to be efficiently specified, too, for example, the maximum number of states to calculate, in order to prevent never ending calculations.

The resulting numbers – normally quite a lot – can be given as tables or graphics. If the concrete choice of parameter values or initial states is not dictated by the situation but just ad hoc in order to be able to start the simulation run, the whole process will have to be repeated with other values fixed – that means defining another scenario – to get an overview over the behaviour of the system in a range of scenarios.

From this description we have a geberal informal definition: By simulation (in mathematics) we understand the effective operational translation of mathematical objects or processes into numerical operations. (Outside mathematics the concept has to be extended to include the building of a mathematical model first.)

Simulation in this sense is a general mathematical method which has always been used but has gained importance enormously through the availabilty of effective numerical machines, especially computers. As a method it is very often not difficult to apply, and it can be a mighty instrument, especially if combined with other, more traditional mathematical methods such as proof, construction, algebraic calculation, analysis, etc.

Here are some examples of simulations:

- Function plotting. The mathematical object is the graph of the function, say of $f(x) = \sin x$, which is a subset of R^2. For the simulation one has first to fix boundaries, say from $-\pi$ to 2π, then approximate the interval $[-\pi, 2\pi]$ by a finite set of floating point numbers, calculate approximations to the sine of these numbers, determine screen pixels to correspond to the calculated values, connect those pixels by the built-in "line-drawing" routines, and display the result. The fixing of parameters will become even more apparent, if you simulate functions with parameters, say $f(x) = \sin(ax), a \in R$.
- Stochastic simulation. The mathematical object is, for example, a stochastic variable with its distribution, mean and variance, say a uniformly distributed variable transformed by some complicated process or function f. To simulate it, you take a finite number of uniformly distributed (pseudo-)random numbers, transform them by (a numerical approximation to) f, take the resulting finite distribution as an approximation to the distribution sought, and calculate its mean and variance.
- Solution to a differential equation.[2] The mathematical object is the general solution to the given differential equation. To simulate it, one chooses several different initial conditions, solves the resulting initial value problems by numerical methods and plots the results. The emerging picture should give some insights into the flow-lines of the differential equation, its overall behaviour and possible locations of critical points.

Simulations normally share a double experimental character: First by the numerical approximations whose errors can be only estimated since the assumptions of strict error control in most cases cannot be verified by numerical methods alone, and second by the fixing of the parameters, boundaries, etc. Simulations need to be complemented by some theory, however rudimentary, in order to lead to insight and understanding. Thus the plotting of the sine-function can only give a non-misleading intuition, if the continuity and periodicity are known or can be abstracted by the consideration of a well chosen sequence of (simulated) pictures with some zooming or similar means. The insight does not come from the pictures. The intellect of the students has to see the connections between the pictures and the necessities behind them; but to see the facts given by the simulation may strongly help the student to understand the facts given by some theory.

Model Building

The building of mathematical models is seen by many people as the heart of application oriented mathematics teaching. If done properly, the usual restriction to linearity assumptions will soon be noticed as inappropriate, and the use of simulation software in order to explore the (mathematical) models developed becomes necessary.

Here we describe briefly dynamic model building of simple growth processes in the mathematics classroom with the program *Modus*, which at the moment is being tested in schools in a preliminary version. As with most dynamic modelling tools, the crucial concepts are the distinction of the main variables as levels and flows. Levels can only be changed through flows; this property is described in formal mathematical language by use of difference or differential equations, the flows being the derivatives of the levels. The model building is done by constructing structure diagrams, thus avoiding the necessity for an abstract formal language. The students easily develop linear and exponential models of growth. The step from linear to exponential growth is made by changing the constant flow to a (linear) function depending on the level, thereby introducing a first feedback loop (see Fig. 12 and 13).

---

[2] An indefinite integration is a special case of this, namely the solution of $y' = f(x)$.

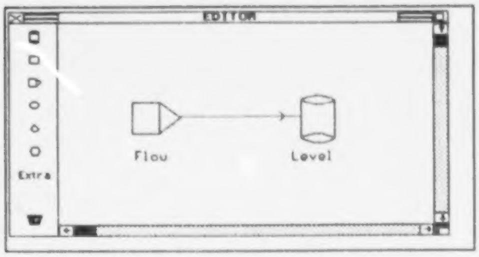


Figure 12

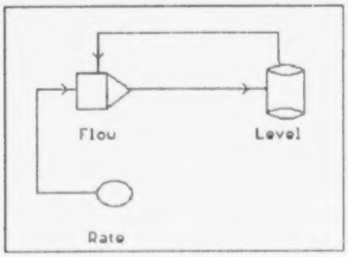


Figure 13

Students soon detect that exponential growth is possible – and widespread – only in the beginning phases of growth processes, be it population growth in a new environment, growth of individuals, development of first love, spread of rumors and epidemics, but growth eventually has to slow down. In model building, this is interpreted as a dependency of the growth rate on the level reached, thereby introducing a new feedback loop, now negative, which yields logistic growth (Fig. 14). Even without any formulas for the logistic function, the growth behaviour can be completely understood from the model itself, and becomes evident by observing parts of the phase diagram being generated dynamically (Fig. 15).

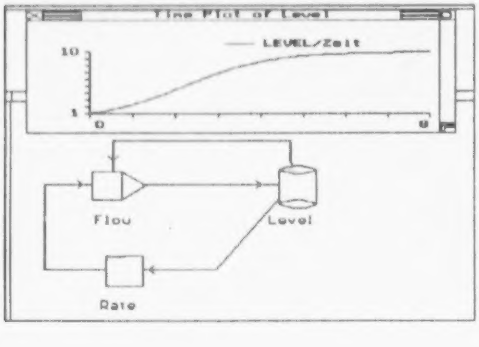


Figure 14

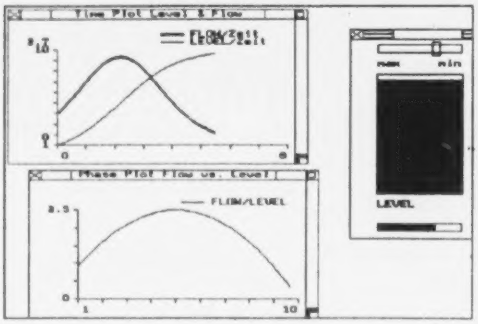


Figure 15

Students (grade 10 or higher) learn to use the argument that from the model, the concrete quadratic dependency of the flow on the level is clear. A positive flow causes the level to grow, which means a motion in the phase diagram to the right, now causing a new flow, etc. The motion in the phase diagram slows down and eventually (virtually) stops, when the flow approaches zero. So the equilibrium can be determined from the model, and similar arguments with starting points above equilibrium show the stability property. Such qualitative understanding has to be developed carefully since it is not easy, but it is more adequate than reasoning with formulas which immediately break down when the model is further refined.

Conclusion

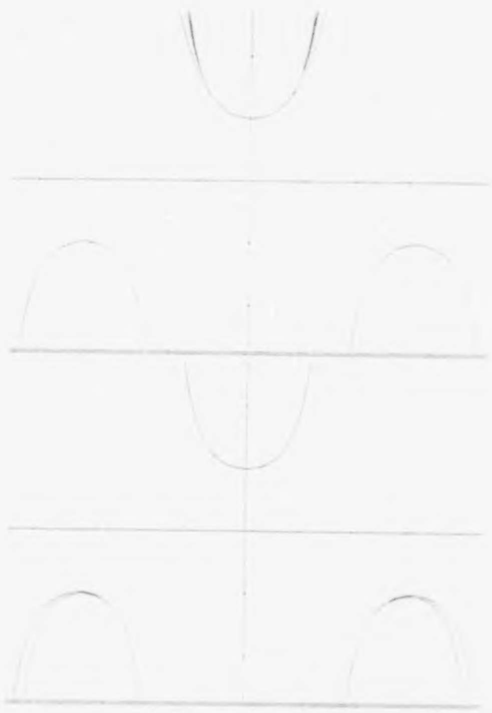
Consequences for Organisation and Curriculum

Organisation problems related to the use of computers in schools have been solved only for demonstration lessons by teachers. Overhead projectors with transparencies for all explications and a special overhead display for all output from the computer instead of the blackboard are standard in many German schools. But 10 terminals are not enough, particularly if students of informatics classes occupy the stations for many hours. Supervision cannot always be done by teachers. The ordinary homework of pupils of math classes using the computer is too easily pushed aside; only pupils with computers at home can help theirselves, provided that their equipment is hardware and software compatible with the equipment of the school. It is difficult too, to organize access to 10 terminals during a written exam for

25 pupils in the class. Therefore, often theoretical questions related to algorithms (additional special cases, restrictions, possibilities for application etc.) have to substitute for the real use of the algorithms for problemsolving in examination periods.

The consequences for the curriculum are very important. School mathematics was determined for centuries by the number of accessible methods for solving problems - equations of no higher degree than 2, systems of equations with 3x3 or 4x4 matrices etc. Application problems were selected carefully so that powerful computational tools were not needed. With the speed and the capacity of memory of modern computers in schools (newer standard: 80386 processors, 1.2 MB RAM), numerical and graphical approximations for solving equations of higher degree and handling matrices of 10 or 20 columns and rows are no problem. Graphical representation of large sets of higher functions or of complicated geometrical situations are also no problem as are the symbolic transformation of complicated rational terms or the symbolic solution of differential equations with interesting initial conditions. With these tools teachers can leave the small garden of traditional school problems and amplify enormously the orientation to modern application.

Let us demonstrate this with two examples. First, from pure mathematics: After teaching curve fitting by Taylor approximations or Fourier approximations in the classical manner with the usual demonstrations you can continue with Padé approximations using rational functions and use these for good approximations to functions with singularities (see Fig. 16).



upper picture: 1/cos(x) approximated by Taylor-serie order 4 and 8 in x = 0 (only the central part of the graph is fitted well)

lower picture: 1/cos(x) approximated by Pade-term order 4,4 in x = 0 (invisible fitting of the curve in the central part and half of the peripherical parts including the singularities)

Figure 16

Our second example is from applied mathematics: The teacher can show how to compute approximations to curves of highways in the student's neighbourhood by parametric splines with the help of the computers.

Thus various new fields are opened for the curriculum. Simulations in natural science and social science using systems of difference equations can be used to solve interesting environmental or economic problems never before accessible in schools. The theory of graphs or the theory of functions with complex variables are other examples of new elementary work with modern tools.

Speculation on the Future

During the first twenty years of computer use in schools, the mathematics classroom was the first place where most students met a computer at all. So math teachers had to pursue an additional goal: Make students familiar with the basic structure and function of a computer system and teach them how to manipulate it. This situation has changed rapidly and will have changed totally in the near future since most students now get acquainted with computers in their daily lives, in their family and recreational environments, perhaps in computer science education, and so on. This means the computer has a new importance in math education, a more fruitful one, more oriented towards mathematics. This is described as follows in a study of mathematics education for the information age to be realized in the Japanese New Mathematics Curriculum [Fujita/Terada 1991]. In upper secondary schools priority should be

> "on giving to students opportunities to do mathematics rather than improving their techniques. Students should understand that computers are powerful tools for intellectual activities by human beings. In this connection, studying mathematics may be the first as well as the best experience for students to use computers for properly intellectual purposes, namely, to study academic subjects with computers. These experiences could even be regarded as a prototype of scientific research activities with computers. Some good students will have chances to observe, to model and to analyze in a mathematical manner various phenomena presented by computers. Furthermore, computer simulation is close to mathematical reality. On the other hand, computers are extremely helpful in fostering students' *mathematical literacy*. Rich mathematical experiences offered by computers, particularly those through operational work by students, will pave the way for the majority of students to grasp concepts and to understand fundamental facts in mathematics."

The New Curriculum plans three courses, Mathematics I - III, in grades 11 - 13 with a total of 10 units (1 unit requires 35 class hours of 50 minutes each), covering a core of mathematics to be learned by all students with Math III certainly to be learned by all science and technology students. Three more courses, Mathematics A, B, C, in grades 11 - 13, totalling 6 units, are composed of four option modules from which two modules are freely chosen for instruction by teachers or schools. Module 4 of Math A, computation and computers, offers students the chance to get to know and become familiar with computers as a tool for mathematics. Module 4 of Math B, algorithms and computers, deals with the powerful function of computers in doing algorithmic computations in mathematics. Math C is characterized by the key phrases "application minded" and "do math with computers" in the areas of matrices and linear computation, various curves, conics and polar coordinates, numerical computation and statistical processing. The study mentions that the newly introduced topics related to computers in Japanese high school mathematics require certain preparation for success, namely, purposeful textbooks, effective teacher training, quality software and relevant development of teaching materials and methods. Indeed, the educational use of computers in class is non-routine and should be explored with respective emphasis of its three aspects; the teacher-initiated use, the student-initiated use and the system-initiated use.

From the viewpoint of a computer-supported curriculum, teaching with computers in a classroom will consist of the following six components:

1) "trial", where learners are invited to the new topic with fun applications offered by the computer.
2) "approach", where learners have heuristic and operational experiences with the aid of computers.
3) "teaching", where the teachers give a lecture and learners get supplementary review and assistance from computers.
4) "experimental understanding", where learners grasp concepts and facts through inductive and experimental recognition with the aid of computers without being burdened by too much drill.
5) "exercise", where learners can perform adequate exercises at their level and using standard (but interactive) CAI.
6) "survey", where learners review the topic which they have learned and are given chances to view further developments and applications.

The principal underlying purpose of the New Japanese Curriculum is to cultivate "mathematical intelligence" by aiming at two targets: Máthematical Literacy and Mathematical Thinking. The aspects from the curriculum mentioned above show that computer systems are considered to be very helpful for both fields.

These two fields are also mentioned among the principles for the development of a new mathematics curriculum in the USA by 2000 [Ralston 1990]. In this reference it is stated that "Mathematical education should focus on the development of mathematical power not mathematical skills". As to information technology there is this principle: "Calculators and Computers should be used throughout the K-12 mathematics curriculum; moreover, new curricula and new curriculum materials should be designed in the expectation of continuous change resulting from further scientific and technological developments". Goals from these principles follow for the elementary grades (1-6) as well as for the secondary grades (7-12). So "the teaching of arithmetic in elementary schools should be characterized by : ... a use of computer software in the teaching and learning process, ... proper and efficient use of calculators for most multi-digit calculations as well as calculations involving negative numbers, fractions and decimals". One important example of computer use in the secondary curriculum follows from the goal that this curriculum should develop students' symbol sense. This means developing "the ability to represent problems in symbolic form and to use and interpret these symbolic representations", and "the ability to identify the symbol manipulations necessary to solve problems expressed in symbolic form and to carry out manipulations using mental computation, pencil and paper, a symbolic or graphic calculator or a computer".

It was noted above that new mathematics curricula should be designed in the expectation of further technical and scientific developments. Most certainly these will occur in artificial intelligence and telecommunications. In a survey on Technology and Mathematics Education, James Fey [1990] writes about artificial intelligence, expert systems and tutors: "One of the very active areas of informatics research is exploring ways that computers can be programmed to exhibit 'behaviour' that simulates human information processing. There are a number of projects in mathematics education that are attempting to capitalize on this computer capability to design programs that act, in various ways, like teachers. The most interesting work along these lines is producing intelligent tutors for an array of mathematical topic areas including arithmetic, algebra, geometry and proof, and calculus. There are some preliminary indications that those tutors provide very effective adjuncts to regular teacher-directed instruction".

As for telecommunications, one might think that this will be important for general or social education only. It is likely, however, that the ability to communicate about mathematical problems in a worldwide group of peers will develop new attitudes towards problem solving, different from the widespread "single attack" of scientists and students. Also, it can be imagined that a feeling for the benefits of international and intercultural understanding can grow more intense through cooperation in a "serious" field like mathematics or science, in addition to the effects of leisure fields like music, movies, etc.

We want to conclude this article by pointing to one of the greatest problems in the changing of the mathematics curriculum under the challenge of computer systems: We must convince the curriculum makers and those who put changes into effect about the necessity and the advantages of this change. We hope that this article will provide good arguments to everybody who wants to tackle this problem.

References

Software

AUTOCALC, Shell Centre for Mathematical Education, University of Nottingham NG7 2RD, UK.

CABRI GEOMÉTRÈ, Baulac, Y., Bellemain, F., Laborde, J.M., Laboratoire L.S.D. IMAG Tour Irma BP 53 X, 38041 Grenoble Cedex, 1988ff. MacIntosh, IBM PC. Versions in other languages such as English or German are available.

DEVELOPING TRAY, Inner London Education Authority, NCET, 3 Devonshire Street, London.

EARLY SCIENCE in *Exploring Primary Science and Technology with Microcomputers*, edited by Jan Stewart, Council for Educational Technology, NCET, 3 Devonshire Street, London.

GEOMETRIC SUPPOSER, Schwartz, J., Yerushalmy, M., et al, Sunburst Communications 1985 ff. Apple II, MacIntosh, IBM PC.

GRAPHIX, Tall, D., van Blokland, P., Kok, D., Duisburg: CoMet Verlag für Unterrichtssoftware 1989. ISBN 3-89418-862-6. IBM-PC. An English version of this program is available' for example, through Sunburst under the title "A Graphic Approach to the Calculus".

KALEIDOSCOPE, Graf, K.-D., Information can be obtained from the author at Freie Universität Berlin, D-W-1000 Berlin 33, Germany.

MODUS, Walser, W., Wedekind, J., Duisburg: CoMet Verlag für Unterrichtssoftware 1992. IBM-PC.

SEEK, Longman, UK; also Shell Centre for Mathematical Education, University of Nottingham NG7 2RD, UK.

SYMMETRIC TURTLES, Graf, K.-D. (see KALEIDOSCOPE)

TIMES in *Teaching with a micro: Math 3*, Phillips, R. et al, 1986, Nottingham: Shell Centre for Mathematical Education. BBC Micro.

Books and Papers

Abelson, H. and di Sessa, A. [1985]: *Turtle Geometry: Computation as a Medium for Exploring Mathematics*, Cambridge: MIT Press.

Biehler, R. and Winkelmann, B. [1988]: *Mathematische Unterrichtssoftware: Beurteilungsdimensionen und Beispiele* in Schmidt, Günter (Ed.): *Computer im Mathematikunterricht*, Der Mathematikunterricht 34, Heft 4, 19-42. ISBN 3-617-24022-4.

Buchberger, B. [1990]: Should Students Learn Integration Rules?, *ACM SIGSAM Bulletin*, 24, 1, 10-17.

Dubinsky, E. and Fraser, R. [1990]: *Computers and the Teaching of Mathematics*, selected papers from ICME-6, Nottingham, UK: The Shell Centre for Mathematical Education.

Fey, J.T. [1989]: *Technology and Education: A Survey of Recent Developments and Important Problems.* Educational Studies in Mathematics, 20, 3, 237-272.

Fey, J.T. [1990]: *Technology and Mathematics Education* in Dubinsky and Fraser, 73-79.

Fujita, H. and Terada, F. [1991]: A Coherent Study of Mathematics Education for the Information Age; a Realization in the Japanese New Mathematics Curriculum, plenary Lecture at ICMI-China Regional Conference on Mathematics Education in 1991, Beijing, China. (Prints available from the first author, Department of Mathematics, Meiji University, Japan).

Goldenberg, E.P. [1988]: Mathematics, Metaphors, and Human Factors: Mathematical, Technical, and Pedagogical Challenges in the Educational Use of Graphical Representation of Functions, *Journal of Mathematical Behavior*, 7, 2, 135-173.

Graf, K.-D. [1988]: Using Software Tools as Additional Tools in Geometry Education with Ruler and Compasses, *Education and Computing*, 4, 3, 171-178.

Graf, K.-D. and Hodgson, B. [1990]: Popularizing Geometrical Concepts: the Case of the Kaleidoscope, *For the Learning of Mathematics*, 10, 3, 42-50.

Johnson, D.C. and Lovis, F. (Eds) [1987]: *Informatics and the Teaching of Mathematics*, Proceedings of the IFIP TC 3/WG 3.1 Working Conference on Informatics and the Teaching of Mathematics, Sofia, Bulgaria, 16 - 18 May, Amsterdam: North-Holland.

Okamori, H. [1989]: *Mathematics Education and Personal Computers*, Tokyo: Daiichi-Hoki Shuppan.

Ralston, A. [1990]: *A Framework for the School Mathematics Curriculum in 2000* in Dubinsky and Fraser [1990], 157-167.

Schumann, H. [1990]: Neue Möglichkeiten des Geometrielernens in der Planimetrie durch interaktives Konstruieren, in Graf, K.-D. (Hrsg.): *Computer in der Schule 3*, Stuttgart: B. G. Teubner, 45-72.

Winkelmann, B. [1987]: *Information Technology Across the Curriculum*, in Johnson, D.C. and Lovis, F. (Eds), 89-94.

Winkelmann, B. [1992]: *Themenheft Wachstum. Dynamische Systeme im Mathematikunterricht*, Soester Verlagskontor, Soest.

A FUNDAMENTAL COURSE IN HIGHER MATHEMATICS INCORPORATING DISCRETE AND CONTINUOUS THEMES

S. B. Seidman
Auburn University, Auburn, AL 36849, USA

M. D. Rice
Wesleyan University, Middletown, CT 06857, USA

THE CURRENT MATHEMATICS CURRICULUM: TRADITIONS AND CONCERNS

For many years, a crucial place in the mathematics curriculum of the last year of secondary school or the first year of university studies has been occupied by the differential and integral calculus. The calculus can be seen both as the culmination of the secondary school mathematics curriculum and as the beginning of serious study of mathematics at the university. In some sense, the study of calculus has become synonymous with the serious study of mathematics. The central and essential position occupied by calculus can be traced to at least two interrelated causes.

For mathematicians, calculus represents the methodology and techniques needed for the study of functions, first defined on the real line, then on higher-dimensional Euclidean spaces, and finally on the complex plane. Thus, the study of the calculus allows students for the first time to acquire the formal abstract tools that are essential for the further study of much of higher mathematics.

On the other hand, calculus provides the foundation for many applications of mathematics to the physical sciences and engineering. These applications date back to Newton's original development of the calculus in the seventeenth century, and since that time they have been wildly successful across a vast collection of disciplines, even including (in recent years), the biological sciences and economics. All of the calculus-based applications are based on mathematical models that can be regarded as being *continuous*; that is, the quantities being modeled are real numbers (or elements of some Euclidean space R^n).

Given both the central mathematical position of the calculus and its vital role in applications (not to speak of the interaction between these two features), it is easy to see why the calculus has occupied such a fundamental and unassailable position in mathematics curricula. During the past several decades, however, the central role of calculus has been seriously questioned, and the questions have been repeated with particular emphasis during the last decade (Ralston 1981, 1989, Kenney and Hirsch, 1991). Just as a major motivation for the predominance of calculus in the curriculum has been the wide range of applications of continuous mathematics, the challenge to that predominance has arisen from the steadily increasing interest in the applications of *discrete* mathematics in many disciplines.

This increasing interest in discrete mathematical applications can be primarily attributed to the widespread use of computers. Computers are essentially discrete machines, and the mathematics that is needed to use them is also discrete. As a consequence, the discipline of computer science is heavily dependent on a wide variety of discrete mathematical ideas and techniques. Furthermore, the easy availability of computers has encouraged the use and development of discrete mathematical models in many disciplines. For example, operations research models (linear programming, integer programming, etc.) are widely used and are based on a discrete mathematical perspective.

It is natural to expect that the rapid growth of interest in discrete mathematics and its applications, fueled by the explosive developments associated with computers, should have an impact on the mathematics curriculum. Although this impact would have been significant under any circumstances, its effect in the United States has been magnified by other questions that have been raised in recent years about the teaching of calculus. Widespread dissatisfaction has been reported with the nature of the calculus courses and the knowledge of the students that have completed them (Lochhead 1983, Steen 1983, Douglas 1986, Steen 1988). The computer is also directly influencing the content of the calculus course itself, both by encouraging the inclusion of numerical methods and by suggesting that symbolic manipulation software may make emphasis on techniques of differentiation and integration obsolete (Bushaw 1983, Wilf 1983, Nievergelt 1987).

In summary, both the nature of the calculus course and the fundamental position that calculus has occupied in the mathematics curriculum for more than a century have come under serious challenge. These challenges have come both from within and outside the community of mathematicians, and they can primarily be attributed to the increasingly broad role that computers are playing in the various scholarly disciplines represented within the university and in the wider world. In the next section of this paper, we will look at the responses that have been proposed to these challenges.

RESPONSES TO THE CHALLENGE OF DISCRETE MATHEMATICS

When any curriculum is confronted by a new topic that should be included, there are essentially two potential responses. The new topic can either be encapsulated in a course that is added to the curriculum, or it can be incorporated as a fundamental constituent of a revised course. Most topics that have been added to the mathematics curriculum in recent decades have been added as new courses (e.g. abstract algebra and topology).

It was therefore natural that when mathematics faculties were asked to include discrete mathematics in the curriculum, this was most commonly done by developing new courses in discrete mathematics. Such courses were designed primarily for students of computer science. There have been two fundamental problems with this approach. First, the discrete mathematics courses were too often taken by third-year students, so that the material was learned too late to be of use in the data structures courses taken by first and second year students of computer science. Second, when students were expected to use their discrete and continuous mathematical skills in fourth-year computer science courses (for example, in the analysis of algorithms), most have found it very difficult to combine these skills effectively. Many students do not see any connections between discrete and continuous mathematics, and are unable, for example, to apply calculus techniques to estimate growth rates of discrete functions or to estimate the size of discrete sums. This inability to combine discrete and continuous skills is also found in students of probability, operations research and signal processing.

Both of the above reasons suggest that discrete mathematics should be incorporated as a component of the fundamental mathematics course that is offered to all students in their first two years of university study. This suggestion was first made by Ralston (1981), who proposed that the study of discrete mathematics *precede* the study of calculus. He argued that such an organization would benefit virtually all students of mathematics, and not just those students concentrating in computer science. Ralston's proposal has led to substantial discussion in the United States on the proper place of discrete mathematics in the curriculum (Ralston and Young 1983). The debate has focused on whether discrete mathematics should *precede* or *follow* the calculus in the curriculum of the first two years. Many of the arguments advanced on either side are administrative in nature, dealing either with the demands of other curricula (such as physics or engineering) or with articulation with other institutions (such as high schools, junior colleges or universities that have retained the standard curriculum). One result of this debate has been the publication since 1985 of over 40 discrete mathematics texts for freshman or sophomore courses (e.g. Ross and Wright, 1988, Maurer and Ralston, 1991).

Whether calculus is placed before or after discrete mathematics, it is by no means clear that students who have completed both courses will be able to combine their discrete and continuous mathematical skills in an effective manner. This problem has been recognized by some designers of proposed curricula, and consequently their calculus proposals generally include some discrete aspects, such as extended discussion of numerical methods and substantial use of sequences (see, for example, Bushaw 1983).

Another possibility, which has been given little serious attention, would be to develop a new, unified curriculum that would interweave discrete and continuous themes throughout its courses. While the first year of the curriculum would correspond to the calculus course, its real thrust would be the study of functional behavior and functional representation. The course would consider discrete functions (sequences) along with continuous functions, and would constantly emphasize analogies and parallels between discrete and continuous situations. Thus the first year of the curriculum would be primarily continuous, but with a strong discrete flavor. The second year of the curriculum would focus on *structure*, and would be primarily discrete, but with a strong continuous flavor.

This paper will argue that a curriculum unifying discrete and continuous themes is not only feasible, but has the potential of providing students with a broad, powerful perspective embracing the mathematical ideas and techniques that are needed for the study of computer science. This perspective would also yield a strong mathematical foundation for the study of engineering, the physical sciences, and indeed for the study of higher mathematics itself.

Furthermore, the development of such a curriculum would force a reexamination of the topics taught in the conventional calculus course. As mentioned above, various recommendations have been made to remove or include particular topics. Although each such recommendation has been cogently argued, no consistent rationale has been given for the collection of topics that together make up the proposed calculus course. The first-year course outlined below has a consistent theme - functional behavior and representation - and each topic to be included in (or excluded from) the course should be judged on the degree that it matches the course's perspective.

In the following section, a detailed outline and discussion will be given only for the first year of the proposed two-year curriculum. At the conclusion of the paper, we shall return to the second year of the curriculum, as well as to the larger issues raised by the question of articulation with other curricula.

A FIRST-YEAR CURRICULUM INCORPORATING DISCRETE AND CONTINUOUS THEMES

The fundamental thrust of the proposed first-year curriculum is the behavior and representation of functions. Roughly, the first semester is devoted to tools for the description and analysis of functional behavior, with the focus shifting to representation of functions in the second semester. Before presenting a more extended discussion of the benefits to be achieved by including both discrete and continuous topics, it will be useful to give an annotated outline of the first semester curriculum.

A. Functions

1. *Number and Relations*

   A knowledge of set concepts and notation is assumed. Inequalities will be emphasized.

2. *Functions and Operations*

   The function concept and functional notation will be introduced, stressing the *algorithmic* interpretation of the function symbol f. Discussion will include domain and range, operations on functions (arithmetic operations, composition, translation), and graphs of functions. Useful functions will be introduced [polynomials, rational functions, exponential functions (defined on the integers), absolute value, floor, ceiling].

3. *Models*

   Algorithms and elementary complexity analysis will be introduced (including binary search). This will allow discussion of the function $\lfloor \lg(n) \rfloor$. Models demonstrating the need to construct functions and to perform curve fitting will be included.

B. Behavior of discrete functions

1. *Sequences: Iteration and Recursion*

   This section will discuss a variety of sequences including geometric squares, the Fibonacci sequence and the sequence generated by the Euclidean algorithm.

2. *Difference Operators*

   The difference operation Δ will be introduced as a function on sequences. The recursion scheme

   $$u_{k+1} - u_k = Au_k$$

   will be treated in order to emphasize special functions defined on the integers. Formulas for higher differences will be discussed.

3. *Summation*

   The primary topic here will be the binomial theorem, both in its standard form and in the expression for $(1+\Delta)^n$. The second form will allow various formulas for finite sums to be presented.

4. *Order Notation (O, o) and Limits of Sequences*

C. Behavior of continuous functions

1. *Limit Heuristics*

   Limits of functions will be discussed only in terms of limits of sequences. The continuity concept will be introduced. The operator

   $$\Delta_h f = \frac{f(x+h) - f(x)}{h}$$

   will be introduced. Analogies to the discrete difference operator discussed above will be pursued.

2. *First Derivative*

   The derivative will be defined, and interpreted using tangent lines. It will be shown that differentiable functions are continuous.

3. *Differentiation Rules*

   Powers and roots; product, quotient rules.

4. *Monotone Functions and Local Extrema*

   A rigorous treatment will be postponed. Curve sketching will be introduced here and the use of graphing calculators will be stressed.

5. *Second Derivative*

   Concavity will be discussed and applied to curve sketching again using graphing calculators.

6. *Extreme Values*

Maximum-minimum problems will be solved. Examples will also demonstrate the use of piecewise linear functions.

7. *Related Rates*
   The chain rule will be presented, and related rate problems will be solved.

D. Estimation and error

1. *Mean Value Theorem*
   Monotone functions will be discussed more rigorously, and the MVT will be applied to global estimation of functions.
2. *Solution of Equations*
   Newton's method will be discussed from both geometric and iterative perspectives. An elementary treatment of error estimation will be given, and critical values will be estimated.
3. *Interpolation*
   Interpolation of functions by straight lines and parabolas will be discussed using the difference operators developed above.
4. *Approximation*
   Second-order Taylor polynomials will be used to approximate functions, and the estimated error will be computed. Analogies will be drawn between interpolation and approximation and between differences and derivatives.

E. Integration

1. *Introduction*
   The summation operator for sequences will be introduced. Its relation to the difference operator will be discussed. It will be treated as an aggregation operator, and used to motivate the discussion of area.
2. *The Definite Integral*
   This will first be introduced using a piecewise linear definition. This definition will then be applied to step functions. The area definition will then be presented, and applied to parabolas using the results on finite sums obtained above. Some elementary properties of the definite integral will be presented, including the mean value theorem for definite integrals.
3. *The Indefinite Integral*
   This will be explicitly computed for step functions, piecewise linear functions and parabolas.
4. *The Fundamental Theorem of Calculus*
   This will be derived from the mean value theorem for definite integrals. The chain rule will be applied to investigate some properties of the integral of $1/x$.
5. *Evaluation of Integrals: Analytic Techniques*
   Substitution techniques will be discussed, as well as the use of integral tables and symbolic calculators.
6. *Evaluation of Integrals: Numerical Techniques*
   The trapezoidal rule and Simpson's rule will be discussed. It will also be shown how integrals can be estimated using inequalities, and how sums can be estimated using integrals.
7. *Applications of Integration: Aggregation*
   The applications to be treated include work and volume.
8. *Applications of Integration: Modeling*
   The primary theme here will be the recognition of Riemann sums in differing situations. Examples will be taken from arc length and fluid flow. The basic point will be that when a model generates a discrete (Riemann) sum, it can then be approximated by a definite integral.

Although this annotated outline gives a good overview of the first semester of the proposed course, it is too brief to show how the interweaving of discrete and continuous themes can lead to major benefits. The following examples are meant to be typical of the perspective that will be possible within this course structure.

Example 1: At the beginning of the course, the discrete exponential function, $f(n) = 2^n$, will be introduced, along with its one-sided inverse, $g(n) = \max\{k|2^k < n\} = \lfloor \lg(n) \rfloor$. The function $g(n)$ is vitally important in computer science; for example, $g(n)+1$ is the worst-case number of comparisons in a binary search of a list of length n. The growth rate of $g(n)$ is important, and is usually treated (via calculus) using L'Hospital's rule. We suggest a discrete approach, based on the binomial theorem. Clearly $2^{g(n)} < n$, so that $g(n)/n < g(n)/2^{g(n)}$. To determine the behavior of $g(n)/2^{g(n)}$ as $n \to \infty$ it is sufficient to consider powers of 2 since $g(n)$ is constant between successive powers of 2. Since for $n = 2^k, g(n)/2^{g(n)} = k/2^k$, it is only necessary to look at the behavior of $k/2^k$ as $k \to \infty$. By the binomial theorem $2^k = (1+1)^k \geq k(k-1)/2$, and hence $k/2^k < 2k/k(k-1) = 2/k(k-1)$, which gives the result that $g(n)/2^{g(n)} \to 0$ and, therefore, so does $g(n)/n$. The simplicity of the discrete argument should aid the student in learning, understanding an assimilating the growth rate of the *continuous* logarithm.

Example 2: The syllabus outline has referred to analogies between the discrete difference and summation operators on the one hand, and differentiation and integration on the other. For example, the difference operator is defined on the sequence $\{u_n\}$ by $\Delta u_n = u_{n+1} - u_n$. If we define the *falling factorial* function on the integers by $x^{(m)} = x(x-1)\cdots(x-m+1)$ then it is easy to see that $\Delta x^{(m)} = mx^{(m-1)}, \Delta^2 x^{(m)} = m(m-1)x^{(m-2)}$, and finally that $\Delta^m x(m) = m!$ and $\Delta^{m+1} x^{(m)} = 0$. Thus the behavior of the difference operator (and its iterates) on the polynomials $x^{(m)}$ is strongly analogous to the behavior of the differentiation operator (and its iterates) on the polynomials $\{x^n\}$. Furthermore, since each collection of polynomials provides a basis for the vector space of polynomials of degree at most n, an example has been introduced which will be useful in a later course in linear algebra.

One further benefit of the use of difference operators is the natural observation that $\Delta 2^n = 2^n$, or more generally that $\Delta k^n = (k-1)k^n$. This suggests that exponential functions, whether discrete or continuous, may have a special role to play with respect to difference or derivative operators, and serves to motivate the later observation that $d/dx(e^x) = e^x$.

Example 3: The first two examples used discrete ideas to motivate continuous concepts that are to be introduced later. In this example, continuous techniques are used to obtain a discrete result. The identity giving the sum of a geometric progression,

$$\sum_{k=0}^{n-1} x^k = \frac{x^n - 1}{x - 1}$$

can be differentiated using the quotient rule to obtain the identity

$$\sum_{k=1}^{n-1} kx^k = \frac{(n-1)x^{n+1} - nx^n + x}{(x-1)^2}.$$

Using this identity, it is immediate that

$$\sum_{k=1}^{n-1} k2^k = (n-2)2^n + 2$$

and that

$$\sum_{k=1}^{n-1} k2^{-k} = 2 - \frac{n+1}{2^{n-1}}.$$

The last result yields

$$\sum_{k=1}^{\infty} k2^{-k} = 2$$

since $k/2^k \to 0$ as $k \to \infty$ (see Example 1). This example serves to remind students that continuous techniques can be important in discrete situations.

These examples demonstrate that the proposed course does not merely insert a collection of important discrete topics into the calculus course, but rather expresses a consistent approach to all of the subject matter. The fundamental perspective is the study of functional behavior, and both discrete and continuous functions are treated throughout. Each class of functions is used to develop tools and suggest analogies that will be useful for the study of functions of the other class.

The second semester of the course further elaborates the functional perspective. Rather than give a detailed, annotated outline, we shall discuss the topics to be covered and describe how they relate to the themes developed during the first semester. The second semester is primarily devoted to material taken from two broad categories, special functions and representation of functions.

Exponential and logarithmic functions will be treated in depth. The natural logarithm will be introduced using the definite integral, and its properties will be investigated. The *inverse* of the logarithm will be motivated using growth models and the differential equation $dy/dx = ky$ and the relationship of this inverse to the exponential function will be motivated using difference equations and the *discrete* logarithm. Finally, the properties of the function e^x will be developed. Numerical estimates for exponential and logarithmic functions will be used throughout the discussion.

The next major topic will be trigonometric functions. Here the primary motivation will come from the geometry of the circle and from models of circular and harmonic motion, although discrete periodic functions, such as mod n, will also be used. The properties of the trigonometric functions will be developed. Integration by parts will be introduced and applied to the special functions. The special integrals leading to the inverse trigonometric functions will be introduced here. Mathematical models suggesting the use of trigonometric polynomials will also be used.

Once the standard functions have been treated, it will be natural to discuss various forms of infinitary behavior. The discussion will begin with a reconsideration of infinite sequences, including a presentation of indeterminate forms and their applications to order notation. The remainder of this section will be devoted to improper integrals and infinite series, emphasizing the analogies between these two forms of infinite summation.

At this point, the focus will shift somewhat from functional behavior to functional *approximation* and *representation*. Thus the next major topic will be power series, with particular emphasis on the use of Taylor series to represent functions. Generating functions for simple recursions will be discussed, and a certain amount of attention will be devoted to computational issues and the estimation of error terms. The constant theme will be the use of Taylor series as function approximations to obtain information about functional behavior that would otherwise be difficult to obtain.

The final topic will be trigonometric series, with particular emphasis on the representation of functions using Fourier series. The treatment of Fourier series at this early point will require the introduction of complex numbers, which will reinforce the students' geometric understanding of trigonometric functions. Furthermore, the availability of Taylor series will permit an analytic as well as a geometric discussion of the identity $e^{ix} = \cos x + i \sin x$. Finally, the early introduction of Fourier series will make it possible to discuss discrete Fourier series and their applications at a far earlier point in the curriculum than is presently possible.

Clearly, the focus on functional behavior and representation has produced a first-year course that is quite different from what is currently taught. The essential core of the current calculus course has been retained, but it is always made clear that it is there because it throws a powerful spotlight on functional behavior and representation.

Conversely, many traditionally taught topics have been removed. This pruning was only possible because the developers approached each topic with the same question: How does this topic impact on the main theme of the course?

Now that the course has been outlined, it remains to show how it will fit into the curriculum. We will also have to pay some attention to the second-year course that will follow this course, and also to the political and institutional problems that its adoption would pose.

IMPLICATIONS FOR THE CURRICULUM

The first question to be addressed is the audience to be served by the proposed course. It is clearly ideally suited for students of computer science, since it merges themes from continuous and discrete mathematics in a synergistic manner. Students who have successfully completed the course can be expected to handle the mathematics arising (for example) in the analysis of algorithms. It can also be argued that this course would be well suited as a first course for students of mathematics, the physical sciences and engineering. For these disciplines the major omission has been vector geometry and multivariate calculus. In many universities, a large proportion of this material is treated in the second year, and it is not unreasonable to suppose that even more could be shifted to a third-semester course designed for those students.

Although much vitally important mathematics can be subsumed under the general heading of "functions", an equally important heading is that of "structure". While the proposed course is intended to give students the most important tools that come under the former heading, it does not address the latter. For students of computer science, both headings are equally important, and thus an important place in their education must be found for "structure". Much of the debate summarized above on the place of discrete mathematics in the curriculum can be seen as a debate on the place of "structure" in the curriculum. Following on the first-year course that has been outlined above, it is reasonable to develop a second-year course focusing on "structure".

Such a course will not be described in detail here, but it is possible to discuss briefly what general topics might be included. The primary strands might be discrete mathematics, linear algebra and probability theory. Discrete mathematical topics could include relations, graphs, Boolean algebras and formal languages. The discussion of linear algebra could include some multivariate calculus, which could then be applied in the probability portion of the course. Just as with the first-year course, the topics included in the second-year course should be chosen because they illustrate vital structural themes or because they are motivated by or permit the development of important applications.

The introduction of courses designed along these lines will not be a simple matter. The obstacles that will be found range from the need for new textual materials to the difficulty of articulating the new courses with other courses and institutions on all levels. It would be an unfortunate mistake, however, to conclude that because of the certainty of encountering what seem to be insuperable obstacles to the introduction of a truly new curriculum, the only possible strategy is one of incremental change. The development and introduction of a curriculum integrating discrete and continuous ideas is an exciting challenge, and one that should be taken up in several places. What is really needed is a collection of design and development experiments, performed in out-of-the-way "protected" environments. Once a new curriculum has proven its viability and worth in one or more of these experimental environments, it will be time to address the structural and institutional issues involved in transplanting the successful curriculum to less protected situations.

REFERENCES

Bushaw, D. [1983]: *A two-year lower-division mathematics sequence* in *The Future of College Mathematics* ed. A. Ralston and G.S. Young, pp 111-118. New York: Springer- Verlag.

Douglas, R. (Ed.) [1986]: *Toward a Lean and Lively Calculus*, Proceedings of a Conference/Workshop at Tulane University, Washington, DC: Mathematical Association of America.

Kenney, M.J. and Hirsch, C.R. (Eds.) [1991]: *Discrete Mathematics Across the Curriculum, K-12*, 1991 Yearbook of the National Council of Teachers of Mathematics, Reston, VA: CTM.

Lochhead, J. [1983]: *The mathematical needs of students in the physical sciences* in *The Future of College Mathematics*, eds. A. Ralston and G. S. Young, pp. 55-69. New York: Springer-Verlag.

Maurer, S. B. and Ralston, A. [1991]: *Discrete Algorithmic Mathematics*, Reading, MA: Addison-Wesley.

Nievergelt, Y. [1987]: The Chip with the College Education: the HP-28C, *Amer. Math. Monthly*, 94, 895-902.

Ralston, A. [1981]: Computer science, mathematics and the undergraduate curriculum in both, *Amer. Math. Monthly*, 88, 472-485.

Ralston, A. and Young, G.S. (Eds.) [1983]: *The Future of College Mathematics*, New York: Springer-Verlag.

Ralston, A. (Ed.) [1989]: *Discrete Mathematics in the First Two Years*, MAA Notes No. 15, Washington, DC: Mathematical Association of America.

Ross, K. A. and Wright, C.R.B. [1988]: *Discrete Mathematics*, 2d Ed, Englewood Cliffs, NJ: Prentice Hall.

Steen, L.A. [1983]: *Developing mathematical maturity* in *The Future of College Mathematics*, eds. A. Ralston and G. S. Young, pp. 99-107, New York: Springer-Verlag.

Steen, L.A. (Ed.) [1988]: *Calculus for a New Century*, MAA Notes No. 8, Washington, DC: Mathematical Association of America.

Wilf, H.S. [1983]: *Symbolic manipulation and algorithms in the curriculum of the first two years* in *The Future of College Mathematics* eds. A. Ralston and G. S. Young, pp. 27- 40, New York: Springer-Verlag.

TEACHER EDUCATION AND TRAINING

Bernard Cornu
Institut Universitaire de Formation de Maitres, Grenoble, France

Introduction

During the ten last years, considerable progress has been made in the development of computer hardware and software, and many valuable educational experiments have been carried on. However, computers are not so commonly used as one might have expected. In many schools the computers are locked in a special room, and it is not easy for teachers to use them. They must plan in advance, be sure the room is available, get the key and check and prepare the computers. Then they go to the computer room with the pupils, and the time spent there is generally not totally "integrated" into the course.

Thus even when the computer is used, the impact on the learning is not clear. For some pupils, it is clearly useful and fruitful, but do we know why and how? We all know very good and enthusiastic teachers using computers, and they generally do it with success. But it is time consuming, it needs a great personal investment, and the conditions of success are not easy to reproduce or to transfer to another situation.

However, computers are now very common in society; they are used in many domains of daily life. In many countries national plans for computer equipment in schools have been achieved, and so a lot of computers are available in schools. Much educational software has been produced, and it is often of high quality. The use of computers does indeed become easier.

Five or ten years ago, the focus was on the development of hardware and software, and on original experiments in using computers in education. Now it appears that teacher training is the next major and unavoidable step but one which has not been sufficiently studied. Most countries are now asking how to train all current and future teachers in the use of new technologies for education.

Of course, training plans have already been tried. The first ones were generally training in computer science. Teachers from various subjects were trained in computer science, and one thought that they would then be able to use computers in their teaching in an efficient way. It did happen but only in some cases! And it did not solve the pedagogical problems of the use of computers, which increasingly appear to be essential.

The use of computers in education has relied mainly on some enthusiastic teachers who spend nights and weekends writing programs and preparing activities for their pupils. These teachers somehow got the training they wished (even if they learnt a lot by themselves!). But we now need to go further, and the way the use of computers was developed with some teachers is certainly not applicable to the teachers. We need to imagine new ways such that all teachers will be able to use computers.

In most countries computer science is not yet a school subject. Therefore, except in some particular cases, we do not need to train computer science teachers, but we need to train teachers in all subjects in the use of computers and new technologies in the teaching of their subject. Thus we need to reflect on the contents of such training.

The main problem, as noted, is the of generalisation. We know how to train some teachers but we now need to train all teachers. We have done some very specific and sophisticated training; we now need training which can be easily generalised and delivered to all. We must take into account the willingness and the abilities of the "standard" teacher, and design adequate training. The usual training for good and enthusiastic teachers is certainly not directly reusable.

This is both a pre-service and an in-service matter. In the next ten years in most countries, one third of the teachers will be changed (because of retirements and the increasing numbers of teachers). So pre-service training will be efficient for this third. The other two thirds will need in-service training during the same ten years.

In the long term, one must think about the link between pre- and in-service training. In an ever changing world, it is impossible to give future teachers the abilities and knowledge they will need throughout their careers. They will have to learn, to think and to reflect continuously. Pre-service training is not intended to avoid in-service training, but, on the contrary, to prepare for it! Increasingly in-service training should be considered a normal part of the job of a teacher. It should not be only for volunteers, but for all!

The evolution of teaching

For several different reasons, teaching is going to evolve:

- Technology is evolving quite quickly. Hardware is becoming smaller and cheaper and more and easier to use. Software is also evolving, becoming more user-friendly, and it is becoming possible to use computers with good software which requires little preparation.

- Pedagogy is evolving. One reason is because research in education provides better knowledge of teaching and learning. And new tools are beginning to be available for teaching. But pedagogy is also evolving because of the democratisation of education. More and more children have access to education, and so pupils are increasingly diverse, and need pedagogy adapted to their needs. In short, teaching needs to be more individualised.

Current and future teachers must be prepared for this evolution. It is not enough to master the knowledge and some pedagogical strategies and tools. Teachers must be able to deal with all the evolution which will happen, and to adapt to many different kinds of pupils.

The school of tomorrow may be quite different. It will be organized according to a variety of pedagogical styles. There will be large rooms for large audiences; standard classrooms; rooms for group work; rooms for individual work; rooms for practical work or workshops; resource rooms etc. Not only will the pupils be provided with a variety of rooms, but so also will the teachers. Teachers now have a rather standard way of working. They come to school to give their lessons, and they stay at home to prepare the lessons or to mark homework. One can imagine that teachers will increasingly work with their colleagues and that they will need to have special tools and materials available for their work. Certainly offices must be provided for teachers, and rooms for group work. They will also need laboratories to prepare lessons using technology.

The school of tomorrow will be equipped with advanced technology – computers, multi-media resources, easily available in each room (perhaps with permanent equipment, or possibly by plugging in portable machines); Resource centers will also be necessary in schools. Libraries with books, software, audio, video, and multi-media products. As is the case already with other subjects , one can imagine that in the near future, mathematics laboratories will be available in most schools.

The role of the teacher is also changing. Since pupils are more and more diverse, the teacher has to intervene in many different ways, not only as a lecturer, giving lessons and delivering knowledge to the pupils. In the classroom teachers must use different pedagogical styles and different kinds of activities. They must also work with small groups of pupils and sometimes individually with pupils. Activities with the pupils may occur not only in the classroom, but also in other rooms of the school such as a resource center, laboratory, a room for small groups etc.

Altogether the teacher has to be a counselor, advisor, organizer, leader and a manager. The task is not only mastering of teaching, but also mastering the management of learning.

The way teachers work every day is evolving. They will probably be in the school all day and all week long and will use various tools in preparing lessons and in teaching. They will work together with colleagues, and even teach together with colleagues. Their personal work will also evolve and be more diverse. The evaluation of pupils is going to be more and more complex, and the role of the teacher in evaluation will be more important. Evaluation itself is becoming more precise and more technical; the use of evaluation in training and in individualization of education will be a major role for the teacher.

Teachers will also have to be involved in the elaboration of pedagogical tools. The evolution of teaching needs new tools, but also new ways in designing the tools. Textbooks, software, video and audio documents and resources for pupils will all have to be better adapted to specific pupils or groups of pupils. Their elaboration will need more techniques, more technology and more professionalism.

Team or group teaching will become more frequent. Teachers will work and reflect together and this will soon be considered as a normal component of the job of a teacher. As intellectuals, teachers must continue training and reflection throughout their professional life.

Thus teaching can no longer be considered only as an art; it is a profession with all the components of the professionalism. And this has consequences for the education and training of teachers. We must train professionals!

A good professional must have access to the best and most efficient tools and must be prepared to use these tools, to choose the tools to be used and to adapt to new tools. Once again we note that continuous training is a natural part of the job of a teacher.

Will the computer be able to replace the teacher? The answer seems to be no. Teaching and learning are very complex processes, and, although technology brings new tools, the main didactic actor is the teacher who manages the learning process and adapts it to each pupil and who insures the socialization of the knowledge and its compatibility with the "external world". Among the productions of the pupils in the class, those must be identified which deserve status of knowledge. Thus the teacher institutionalizes knowledge.

Of course, personal and individual moments are possible and necessary in learning, and the computer can make them more efficient As well, some particular topics can be learned "automatically" with a computer (for example, typing; training in repetitive skills such as computation, learning "by heart", etc.). But teaching and learning need interaction between the teacher and the pupils, and among the pupils; the computer must be used in a way that facilitates this interaction.

Teacher training

Which competencies?

In talking about teacher training we need first to determine the competencies which are necessary for a teacher. Which teachers do we need for tomorrow? Which kinds of teachers do we want to prepare?

First, we certainly need teachers who have mastered perfectly the knowledge they will have to teach; teachers *must* be competent in their subject. But this is not enough. They must not only be good "in" their subject; they must also be good "about" their subject. They need to know about the origins and evolution of their subject, about its history and epistemology. They need to know about the role of their subject in society and about its applications. They need to know about the "philosophy" of their subject.

We need teachers who are able to communicate knowledge and to make pupils construct their own knowledge. Teachers must be educated in the pedagogy and didactics of their discipline. They must know about the obstacles to learning, they must know about the errors students may make and their role in dealing with these errors. They must know about the conditions which facilitate learning and they must know about evaluation.

We need teachers able to manage and lead their classes. They must know about groups and individuals. They must have some knowledge of psychology.

We need teachers able to advise and orient their pupils. Thus they must know the educational system, its place and its role in society so they need some knowledge of sociology.

We also need teachers trained in the technical aspects of their job, able to speak loudly and clearly enough and able to use technical tools etc.

In general then, teachers have many different roles and must be competent in each of them.

But what about new technologies and computers? They are linked with each of the aforementioned competencies. One must think about the role of the computer with respect to the subject itself. What is the influence of the computer on mathematics, on the way mathematicians work and on the mathematics which is taught in schools? What is the place of computers in the way mathematics is used in society? What is the influence of computers on the pedagogy and didactics of mathematics? On evaluation? What is the role and the use of the computer in class management, in individualization, in the organization of the teaching? How does it affect the psychology of the pupil? What technical help can the computer bring to the teacher?

Of course, there are no definite answers to these questions; the education of teachers must make them able to ask these questions and reflect about them. Education cannot give definite competencies, but it must give an aptitude to evolve; it must give the basic tools necessary to be able to build one's own strategies, one's own answers.

We now try to list some of the competencies a teacher needs in computers and computer science, remembering that our purpose is not to train computer science teachers, but mathematics teachers.

- Basic tools: such as word processing, spreadsheets, data processing, and also other technological tools such as video and the overhead projector. This is certainly a very important point: If we want ALL teachers use new technologies, they must be totally familiar with the most common and easy to use; it is an absolute necessity that teachers be able to use computers for elementary applications. This is the way to make the computer part of the "daily life" for teachers.
- Technical elements: to be able to use the hardware, to manipulate the main accessories, to identify elementary troubles, and to deal with the technology in the school; teachers need a basic level of "familiarity" with technique.
- Elements of computer science. But to what extent? Teachers certainly need to know just enough in order to "understand what happens"; but the links between mathematics and computer science are so strong that it is certainly useful to know about some fundamental concepts (as well as some concepts of algorithmics - see the chapter by Maurer).
- Mathematics and informatics. Mathematics is evolving and changing under the influence of computers and informatics. Therefore, teachers need to maintain their mathematics knowledge and to practice mathematics from an informatics

viewpoint. Mathematics is becoming more experimental, more algorithmic, more numerical; teachers must be able to follow the evolution of mathematics, and to acquire new competencies and new attitudes and to be able to carry out new activities in mathematics.

- Using existing resources. Teachers must be able to know what exists – different software, different tools, different strategies for teaching. They must be aware of new products which appear. They must be able to choose among existing resources according to the needs of their pupils and according to their pedagogical choices. They must be able to advise pupils which products they should use.
- Pedagogy, didactics and the computer. One of the main problems of the use of computers for mathematics teaching is the integration of the computer activities into the pedagogical strategy. Too often, computer activities are just added to the usual lessons. An optimal use of the computer needs not only good knowledge of the hardware and software to be used, but also mastering of the problems of learning. A teacher should be aware of what we now know about how pupils learn; the computer should be just a tool to implement new strategies and new solutions to learning problems. It can be an efficient tool, for example for individualization of the learning and for evaluation, but only if individualization or evaluation problems are solved in pedagogical and didactical terms. Technology does not replace pedagogy. So, training in new technologies cannot be independent of training about pedagogy and didactics.
- Didactical engineering. Teachers have to elaborate the situations needed for pupils. Since they have a large number of tools at their disposal and a large number of choices in terms of strategy, a teacher needs to have the characteristics of a "didactical engineer", i.e. they must have the ability to use the results of research or theoretical statements and transform them into usable products.

Which methodology for training?

The methods used in teachers training are at least as important as the contents of the training. It is well known that teachers usually teach, not as they were taught to do, but by reproducing the way they were taught. If you only use lectures to train teachers (even if you lecture about active methods for teaching), they will then mainly give lectures to their pupils.

So the most important thing in educating teachers how to use computers in teaching is not to give lectures on "how to use computers", but to actually use the computer in the training. This is true for all new technologies. You should use the overhead projector in the training, rather than give a lecture on "how to use the overhead-projector".

If you want to convince teachers that pupils can learn better with the computer, just make these teachers or future teachers actually learn something with the use of computers.

This means that the training should include active parts, even if some theoretical aspects are also necessary. One often says that problem solving is a good way to learn mathematics; similarly, the solving of teaching or learning problems is a good way to learn about pedagogy, and the solving of teaching or learning problems using new technologies is certainly a good way to learn about the use of new technologies in education.

Teacher training should not be only an accumulation of knowledge. As already noted, teachers should be prepared to evolve and adapt to new situations.

Among the different methods which can be used for training teachers, "training by research" is probably one of the best. It does not mean that all teachers should be researchers. But they should be able to use the methodology of research, and this can be learnt through group activities – reflection, innovation, preparation of documents and of situations, etc. Teachers will need to learn to work in teams with colleagues. To be prepared for such activities, they need team activities in their training!

Teachers should also be trained to communicate, to read, to write (for their pupils; for their colleagues; for publication) since this will also be a component of their job.

Teachers should be prepared for a diversity of pedagogy. There exist many different pedagogical strategies, many different pedagogical styles. Too often, one is convinced that one of these methods is the best. But it is better to be able to determine, in given conditions, at a given moment, with given pupils what is the appropriate method to enable them to learn a specific topic. This implies that in the training itself many different methods and strategies will be used – lecturing with one computer in the room, used mainly by the teacher ("blackboard computer"); collective activities in a room with one computer for each student or group of students; individual activities on computers; self-evaluation using computers etc.

Diverse software must also be used in teacher training – utilities, basic tools such as word-processors, languages, tutorials, open-ended software, multi-media tools etc. Giving teachers access to the maximum of diversity increases their freedom in their own professional activities.

Which contents?

This is again a question without a definite answer. Of course, the answer should not be the same in pre- and in-service training. But, in fact, the topic of computers is new for most in-service teachers, and they need training which is close to that in pre-service.

We can hope that in the future students will have acquired the necessary elements about new technologies in their previous studies, for example using computers for word-processing.

The first question to be asked is: Do the teachers I train need computer science? Will they consider computer science as a subject in itself? Should they only learn informatics as it impinges on their main subject, and through its use in their subject?

Many different answers to these questions have been attempted. Some countries have tried to train teachers by giving them a full year of training in computer science. This produces "specialists", but the reinvestment for other teachers was not easy. Many countries organize sessions for teachers. Here again the diversity of what is offered to the teachers is certainly a good thing – lectures on specific topics; one week or two weeks sessions; a course over one term or one year etc.

In pre-service training, there should certainly be specific modules in order to prepare future teachers for the use of computers. As a possible example, here is the contents of a course we have given for many years at Grenoble University, both to future teachers and to in- service ones. This course lasts for 150 hours (5 hours a week during 30 weeks). The title is: "Informatics for mathematics teaching". Each week, 2 hours are devoted to lectures and 3 hours to practical work. The course is divided into three parts:

- Informatics and algorithmics. Students learn the basic use of a computer; they also learn elements of algorithmics, including recursion, proof of programs, evaluation of algorithms and data processing. They use three different languages for programming: Pascal, Logo, and Prolog.
- Mathematics from an informatics viewpoint. In order to use the computer in mathematics teaching, it is necessary to use it for mathematical activities, and therefore to reconsider some mathematics concepts with the help of computers. Every year in the course we choose different mathematics topics in the curriculum of university studies (not in the curriculum of secondary schools because we want the students to be able to accomplish by themselves the transfer of these activities to the field of secondary school mathematics).
- Pedagogical and didactical viewpoint. In this part, we use and analyse various existing tools (software, textbooks with computer studies integrated into them etc.). We try to combine the fundamental notions of pedagogy and didactics of mathematics together with technology. We also try to make the students solve teaching problems using computers. (For example: I must prepare a lesson about linear equations for tomorrow. What will the content of the lesson be? What software will I use? What will be the activities of the pupils? Here is another example: I must prepare a course about linear equations, but I have six months to prepare it. How will I do it?).

During the year the students have to produce a personal project which takes the form of a piece of software they design and experiment with.

Research, innovation and training

The development of the use of new technologies in mathematics teaching makes it necessary that research be carried on in several domains – research about mathematics learning; research about computers in mathematics teaching; research leading to appropriate software; research about teacher training. This research may take several forms – fundamental research, applied and experimental research, innovation. Too often, there is a gap between the fundamental results of educational research, and products which are usable in teaching and in training. We need to develop applications and implementations of the results of the research and we need pedagogical products based on research.

Innovation and research can contribute to teacher training. Indeed, the participation of teachers or future teachers in innovative activities is a good way for training them.

The participation of teachers in elaborating and experimenting with pedagogical products is necessary, but not sufficient. Designing good software needs computer scientists and software specialists as well as specialists of pedagogy and teachers (practitioners). It is a professional matter which needs professionals.

Two other tracks need to be explored:

- Teacher training is becoming more complex, and we need courses and training activities adapted for this purpose. Courses for teachers and future teachers must be developed. We also need to reflect about the specific competencies which a teachers trainer must have. In fact, in many countries the first problem to be solved before we are able to train ALL the teachers is to train teacher trainers.
- In order to diversify the tools usable in teacher training, it would certainly be interesting to develop computer tools and software for teacher training.

Conclusion

We have done a lot of experimenting with new products and new strategies, and the most enthusiastic teachers have shown both their efficiency and their limits. The problem now is to generalise the use of new technologies, so that ALL teachers are able to use them as they wish, or to know why they do not want to use them.

Two conditions seem to be essential in order to help all teachers and future teachers use computers:

- Make the computer actually available and usable; make it a "daily life" tool; make it really user-friendly. This means that it is necessary that schools be well equipped. The aim should be that each teacher or future teacher has a computer (either one the teacher owns (special plans may need to be set up for purchasing computers at reasonable prices; loans may need to be obtained for future teachers) or the institution must own computers and make them available for teachers and future teachers)
- Actually use the new technologies in teacher training, and not just train about "how to use" them.

In no case can the technology replace the pedagogy. A bad teacher using computers will certainly still be bad! So training and education are necessary, but not only from the viewpoint of technology. We need coherent training, integrating both technological and pedagogical approaches. Teachers must be ready to evolve and adapt, and must retain the ability to ask questions. At each instant they should ask whether education or technology is the driving force.

Teacher training is a continuous process. Preservice and in-service training are strongly linked, and both are necessary. No longer can a teacher be provided with all the abilities and knowledge needed at the beginning of a career; training never ends, reflection never ends; in-service training should be considered as a natural component of the teacher's job. We must never forget that teachers are professionals, and need professional training.

Many countries use a "cascade model" for teacher training. The education ministry organises a course for a number of selected trainers; afterwards, each of them trains a number of other trainers, who then train teachers (or trainers who train ...). Such a model can be efficient, but may also not be! The main characteristics of good training – motivation, activities, understanding – must be present at each stage of the "cascade". And this model can apply only to very specific, precise, and limited training.

"Training plans" have been set up and implemented in many countries for training teachers in the use of computers. They have only partly succeeded. One reason for this is that they are generally too restricted as to technology. A training plan should be more global, aiming not only at solving new technology problems, but aiming at solving teaching and learning problems. New technology problems should not be treated in too isolated a context.

REFERENCES

Ball, D. et al (Eds.) [1987]: *Will Mathematics Count? Computers in Mathematics Education*, an AUCBE report.

Bosler, U. and Squires, D. [1989]: *Training teachers to design educational software* in *Educational Software at the Secondary Level*, (Tinsley, J. D. and van Weert, T. J., Eds.), Amsterdam: Elsevier.

[1988]: *Pre-service teacher education*, Proceedings of ICME6 (Hirst, A. and Hirst, K., Eds.), Budapest: J. Bolyai Math. Soc.

[1988]: *Information Technology in Computer Education*, IBM Denmark.

[1990]: *The Teacher Today: Tasks, Conditions, Policies*, OEDC.

THE IMPACT OF SYMBOLIC MATHEMATICAL SYSTEMS ON MATHEMATICS EDUCATION

Bernard R. Hodgson
Université Laval, Québec, Canada
Eric R. Muller
Brock University, Ontario, Canada

Symbolic manipulators, that is, computer programmes with the capability of carrying out symbolic computations, for example, in calculus or linear algebra, are now widely available. While these are well-established tools in many areas of mathematics, science and engineering, it must be recognized that they are still in their infancy with respect to their use in mathematics education. They represent an ineluctable challenge to current approaches to the teaching of mathematics and there is a belief among some members of the mathematical community that electronic information technology, through these symbolic capabilities, will exert a deep influence on how and what mathematics is taught and learned (for example see Page (1990)). However no clear pattern has yet emerged on how such an influence is to be articulated.

This paper will discuss certain aspects of the impact of symbolic manipulators on mathematical education in the upper secondary years and the first few years of university. It is by no means intended to give the final word on such a vast field as much work is in progress and the technical environment (computer hardware/software and calculators) is constantly improving. The aim of this paper is rather to examine some of the major issues and to indicate general trends which have developed since the 1985 ICMI Study on "The Influence of Computers and Informatics on Mathematics and its Teaching". The influence of symbolic manipulators on more advanced (senior) mathematics courses will not be explored. This is not intended to belittle their impact at this level but rather to concentrate on those years where these systems must be implemented in order to benefit the largest possible number of students in mathematics courses. The influence of these systems and their mathematical foundations (see for example Davenport, Siret and Tournier (1988)) will be thrust into the upper level courses by more capable and interested students as they progress through the system.

Section 1 defines Symbolic Mathematical Systems in broad terms and presents an example of their potential use in mathematics education. Section 2 raises some general concerns related to the impact of these systems on mathematics education while Section 3 discusses implementation of some of the required changes in secondary and university mathematics education. The Appendices provide the following additional information: (1) references dealing with the technical aspects of some of the better known Symbolic Mathematical Systems, (2) further illustrations of the capabilities of these systems, and (3) references to current projects aimed at the integration of such systems into mathematics education.

1. Symbolic Mathematical Systems

The term Symbolic Mathematical Systems is used to define calculator and microcomputer systems which provide integrated (1) numeric, (2) graphic, and (3) symbolic manipulation capabilities[1]. *Numerical* computations have always been included in the domain of both the calculator and the computer. This capability is usually thought of as the ability of doing decimal arithmetic. For example, if $1/3 + 1/9$ is input, then the approximate solution 0.444444 (to some prespecified number of digits) is provided. Symbolic Mathematical Systems have the ability to perform rational arithmetic, that is, to give the exact answer $4/9$ if the input is $1/3 + 1/9$. The user must request the decimal approximation if it is desired. *Graphing* is a more complicated numerical activity. Calculators with graphic capabilities (for example the Casio fx-7000G, Hewlett-Packard HP-48SX or Texas Instruments TI-81) as well as microcomputer graphing programmes are available. To many mathematicians and mathematics educators ***symbol manipulation*** by calculators (for example the Hewlett-Packard HP-28S or HP-48SX) and microcomputer programmes (for example Maple, Mathematica, Derive to name a few) was a most unexpected development. It is the one capability which has the potential of producing the most radical changes in the teaching of mathematics at the secondary school and university levels.

---

[1] It should be noted that in a much more general context, the expression "symbolic computation" could be construed as referring to various types of symbolic objects, for example as described by Aspetsberger and Kutzler (1988): geometric objects (***computational geometry***), logic objects (***automatic reasoning***), programmes (***automatic programming***). The concerns of this paper are limited to computations involving algebraic expressions, so that typical topics of the field are symbolic differentiation and integration, calculation of sums and limits in closed form, symbolic solution of systems of equations and of differential equations, polynomial factorization, manipulation of matrices with or without numeric entries, arbitrary precision rational arithmetic computations, etc. These are sometimes misleadingly called "Computer Algebra Systems" — but they can do much more than algebra as will be illustrated by the examples in this article.

To convey a feeling for some of the capabilities of Symbolic Mathematical Systems and how they could be used in a calculus class, consider the following example of a session with a specific system (namely Maple but this particular choice is not crucial). Such an example could be done in class, or could be structured as part of a laboratory exercise. The example illustrates the numeric, graphic and symbolic manipulation capabilities of the system and shows the system can be used in a mode which requires no programming by the user, but only the knowledge of a few command words. For ease of understanding lines starting with a # (and in *italics*) are external comments, lines starting with a ::: are the user's input and the lines in **bold** are the (Maple) system's response.

#The task is to explore the derivative of ln(x) using
#the definition of the derivative. First the limit of
# (ln(t)−ln(4))/(t−4), called y, as t approaches the
# integer value 4 is explored from a numerical
#point of view, by computing the value of y around
#t = 4. Clearly the value at t = 4 does not exist.

::: y:= (ln(t)−ln(4))/(t−4);

$$\mathbf{y := \frac{\ln(t) - \ln(4)}{t - 4}}$$

#At t = 3.99

::: subs(t=3.99,y);

$$\mathbf{-100\ \ln(3.99) + 100\ \ln(4)}$$

#Evaluation using floating-point arithmetic of this
#last displayed expression then gives

::: evalf(");

.250313

#At t = 3.999

::: evalf(subs(t=3.999,y));

.250031

#At t = 4.01

::: evalf(subs(t=4.01,y));

.249688

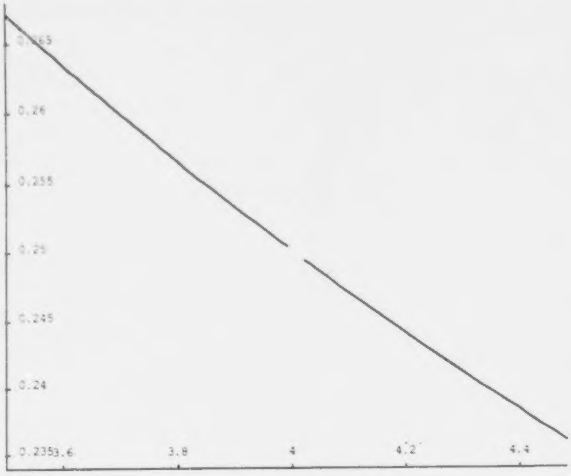
#At t = 4.001

::: evalf(subs(t=4.001,y));

.249969

#Looks as though the function is approaching 0.25
#as t approaches 4. Does the graph support this?
#A plot of y for 3.5 < t < 4.5 is obtained.

::: plot(y,3.5..4.5);



#Yes it does and the graph indicates by a hole that
#the function is not defined at t = 4, where y is
#approximately equal to 0.25. One repeats this
#experimentation with a few more integer and
#rational cases, for example 5, 3/2, 7/3. Then the
#symbol manipulation capabilities can be used to
#evaluate the limit directly.

::: limit((ln(t)−ln(3/2))/(t−3/2),t=3/2);

$$\mathbf{\frac{2}{3}}$$

#suggesting that the limit of
#(ln(t)−ln(a))/(t−a) as t approaches a is 1/a
#for all a>0. This is confirmed by the system.

::: limit((ln(t)−ln(a))/(t−a),t=a);

$$\mathbf{\frac{1}{a}}$$

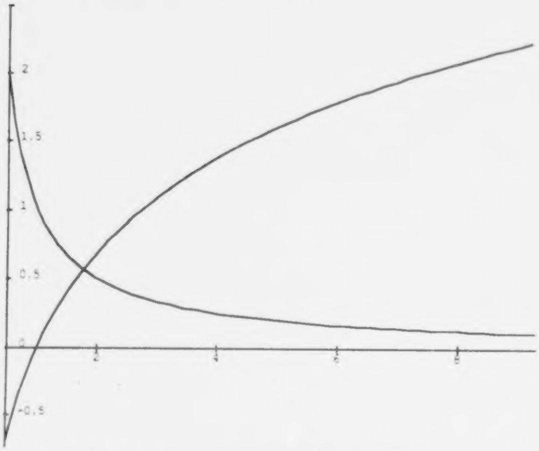
#Which is also confirmed by the differentiation
#capability of the system.

::: diff(ln(a),a);

$$\mathbf{\frac{1}{a}}$$

#Does the derivative have the properties expected?
#Plot the function and its derivative on the same
#graph (in some judiciously chosen interval !).

```
::: plot({ln(t), 1/t}, 0.5..10.5);
```



#Yes ln(t) is a monotonically increasing #function and the derivative is shown to be #positive in the chosen range. Measurements along #the axes appear to confirm the previously #computed points (4,1/4) etc. The formal #definition of the limit can also be explored, namely, #L is the limit of y as t tends to a if for every #eps > 0 there is a del > 0 such that if #0 < |t−a| < del then |y − L| < eps. #Consider the case explored earlier where it was #conjectured that the limit of y as t tends to 4 is #1/4. Select eps = 0.01 > 0; is there a del such #that for 0 < |t − 4| < del, then |y − 1/4| < #0.01? The condition |y − 1/4| < 0.01 can be #rewritten as 1/4−0.01 < y < 1/4+0.01. #This is solved using the system. (Maple solves #different types of equations: algebraic, numeric, #differential, etc.) In this case we are #interested in the numerical solution of an #equation in one variable. Numerical procedures #for the solution of such equations often require #the user to specify an interval within which one #expects to locate the root. In this particular case #Maple does not require such a prompt and #provides the following:

```
::: fsolve(y=0.24,t);
```

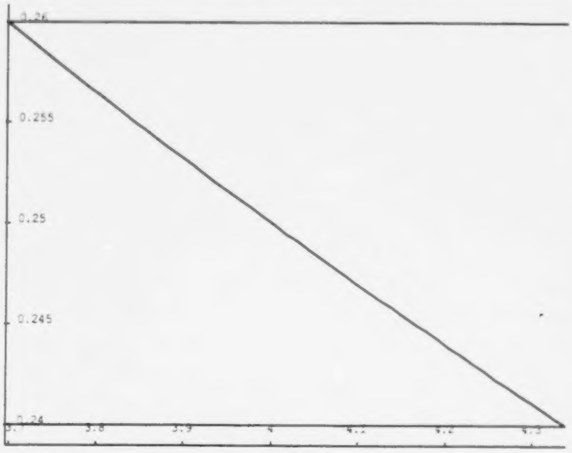
$$4.33789986$$

```
::: fsolve(y=0.26,t);
```

$$3.696303966$$

#From these two values it is concluded (based on #the continuity of the log function) that there is a #del, for example 0.2, such that when #0 < |t − 4| < 0.2, then y is in the specified #range. This is visualized with the following plot, #where we notice that the graph of y is #completely contained in the specified window.

```
::: plot({y, 0.24, 0.26}, 3.6963..4.3379);
```



#To demonstrate how incredibly sensitive and #accurate the limit procedure is, one can consider #the following.

```
::: limit((ln(t)−ln(3.2))/(t−3.2),t=3.2);
```

undefined

#What happened? To resolve this apparent #anomaly the user must realize that elementary #functions involving numbers other than integers #or rationals are approximated (the calculator #mode), that is ln(3.2) is evaluated as shown by #the following output

```
::: y:=(ln(t)−ln(3.2))/(t−3.2);
```

$$y := \frac{\ln(t) - 1.163150810}{t - 3.2}$$

#and, because of the numerical approximation, the #limit of y as t approaches 3.2 does not exist.

For those who are not familiar with Symbolic Mathematical Systems Appendix 2 provides further examples of their capabilities. While special purpose packages have been created to cover specific aspects or topics within the mathematics curriculum, this paper is concerned with "full service" Symbolic Mathematical Systems which can become part of mathematics education across different courses and at different levels. The more powerful systems were originally created to help individuals perform complicated yet algebraically routine mathematics. There is no evidence that the introduction of inexperienced students to more dedicated (smaller specially developed systems addressing one part of the syllabus) has been more successful than introducing them to the larger more sophisticated systems.

2. Mathematics Education Concerns

Mathematics educators must continually make decisions about what mathematics is to be taught, how it is to be presented and what student activities are to be required or encouraged. To this decision making must now be added the role of Symbolic Mathematical Systems. These systems are a fact of life and can no longer be ignored. Mathematics educators have the responsibility to decide consciously whether this environment is to be included within the student's educational experience and what should be the exact role of the Symbolic Mathematical System. This decision cannot be taken lightly for these systems can perform all the mathematical techniques presently included in secondary school mathematics programmes and most of those included in the first two years of university mathematics. The decision to include or exclude the experience of a Symbolic Mathematical System has far reaching implications to the *student*, the *teacher* and to the *curriculum*. These are now considered in turn.

a) Implications to the student

The magnitude of the experiences promised to the *student* by Symbolic Mathematical Systems is illustrated by the following allegory:

A person explores her surroundings by walking (pencil and paper) — many interesting things are discovered, but situations in the neighbouring province are too far away to be experienced, so the use of a car (standard scientific calculator) is allowed. As she drives along, local attractions are overlooked in order to get to her destination. However, even with this mode of transport, she cannot explore distant lands, so an airplane (Symbolic Mathematical System) is provided. She lands in a country where the language is not her own, customs are different — as educators we would try to prepare her for this shock — but there is nothing that is quite like being there. What potential benefit awaits her! — she can now explore concepts which were unknown before and she can contrast, compare and have a different view and appreciation of her own culture and home environment. In this new land she continues to use the other modes of transportation, namely, walking and driving to enhance her experience.

Mathematics education has many of the properties of this allegory. Individuals develop their mathematical understanding in various ways. Due to the different roles played by the left and right hemispheres of the brain, it is most likely that the representation of mathematical concepts in complementary modes such as numeric, graphic, and symbolic will enhance the learning process. For the first time in the history of mathematics education Symbolic Mathematical Systems offer the ability to move easily and rapidly between these different representations. It is expected that the use of paper and pencil will be retained by most students; however, one should not be surprised to find students who can operate completely within the computer environment since most systems now provide for easy interplay between word processing and Symbolic Mathematical Systems.

b) Implications to the teacher

For the *teacher* Symbolic Mathematical Systems are remarkable not only because they can be used to directly perform rational, symbolic or graphic computations but, more importantly, because of what they suggest about mathematics itself and about mathematics teaching. As Young (1986) puts it, "(...) we are participating in a revolution in mathematics as profound as the introduction of Arabic numerals into Europe, or the invention of the calculus. Those earlier revolutions had common features: hard problems became easy, and solvable not only by an intellectual elite but by a multitude of people without special mathematical talents; problems arose that had not been previously visualized, and their solutions changed the entire level of the field." Symbolic Mathematical Systems are part of this revolution. They can serve to help concept development and, by permitting easy and efficient processing of non-trivial examples, they can stimulate exploration and search for patterns[2], generalizations or counter-examples. The teacher must now question the whole of mathematics education. For example, it is increasingly difficult to justify wanting students to become good symbol manipulators unless it can be shown that such procedural skills are essential to an understanding of the underlying mathematical concepts — but no one has yet so shown. However this does not imply that students no longer need develop "symbol sense" —just as the arithmetic calculator has not reduced the need for "number sense". Suddenly the teacher is brought to question both the *content* of the mathematics courses and their *presentation*. As the former relates to curriculum more directly, the latter concern is addressed first.

---

[2] "The rapid growth of computing and applications has helped cross-fertilize the mathematical sciences, yielding an unprecedented abundance of new methods, theories and models. (...) No longer just the study of number and space, mathematical science has become the science of patterns, with theory built on relations among patterns and on applications derived from the fit between pattern and observation." Steen (1988).

The teacher must consider many factors which affect the learning of mathematics. An important factor is the social environment. Some students find it easier and more enjoyable to work on their own while others prefer to work in groups. Some depend on the verbal or written or visual presentation of mathematical concepts by those who understand them. Others find this distracting and prefer to work directly from books. Computers provide opportunities to enhance these social environments. They also introduce a new factor — the computer — which may, for some individuals, erect new barriers and difficulties. It is therefore important for mathematics educators to provide alternative environments for students to experience. Individuals will then be in a position to evaluate them and decide which provide the most opportunities for the development of their mathematical knowledge.

Symbolic Mathematical Systems can be integrated into mathematics education in a number of different ways. The three most obvious ones are:

(1) The teacher can use it as part of a *lecture* or *class presentation*. This requires some projection facilities to allow the students to see what appears on the computer screen. For the mathematics instructor the use of such a system in the classroom provides very different class dynamics. Attention has to be paid to typing, errors, unexpected forms of expressions, graphs which appear different from the traditional book presentation (cf. Muller (1992)), multiple answers, etc. Many mathematics instructors find this situation difficult to handle. Perhaps the central aspect in the successful integration of a Symbolic Mathematical System in the classroom is a necessary evolution of the role of the teacher where intervention is no longer restricted to exposition. Instead the teacher must become a "facilitator" creating a context appropriate for a fruitful interaction between the student, the machine and the mathematical concept. The lecture-examples format must be replaced by a more open-ended approach. Although such a point of view is desirable even in a computer-free classroom, it becomes essential when computers come into play. One of the reasons why films and videos have played such a small role in the mathematics classroom may be the mathematician's belief that you understand mathematics by doing it and not by viewing it. Unlike film, Symbolic Mathematical Systems provide an active environment requiring constant intervention and change of direction. Nevertheless it would be naive not to realize that many teachers will find the sacrifice of traditional security quite threatening. This will be especially true of mathematics teachers who see their role as one of "professing" well-polished mathematical knowledge. White (1989) has suggested that the use of Symbolic Mathematical Systems "can be assimilated most easily in traditional teaching methods and curricula." However, in practice, finding an appropriate role for the teacher may prove to be a major barrier for the universal introduction of Symbolic Mathematical Systems into the traditional lecture presentation and teachers should seriously look at alternative and/or complementary modes of implementation. Even though introducing an occasional Symbolic Mathematical System demonstration into a traditional set of lectures is a start, what is needed is a complete rethinking of the objectives of those lectures.

(2) The technology can also be used in scheduled *laboratory sessions*. This is probably the least threatening mode of introduction for the teacher. Laboratory activities can be developed and tested before the students try them. Students can be given materials to prepare for the laboratory sessions and support can be provided for the students during their scheduled laboratories. The physical laboratory setup can vary. There are advantages to having students working with their own system and advantages to having four to six students working together with a single system. Activities appropriate for laboratory work with a Symbolic Mathematical System should not be a simple duplication of activities which can be achieved just as easily with pencil and paper. What are appropriate activities? Clearly the lack of sustained experience limits one's vision. Nevertheless it is suggested (cf. Muller (1991)) that laboratory activities should meet one or more of the following general attributes:

(a) they encourage exploration of mathematical concepts;

(b) they probe inductive reasoning and/or pattern recognition;

(c) they investigate interrelationships between different representations - algebraic, graphical, numerical, etc.;

(d) they involve problems which would be very difficult and/or too time consuming to solve without the technology.

One can visualize a situation where the *lecture* and

laboratory activities are merged and the lecture presentation takes place in an area where students have access to systems. Because students work at different rates with systems it is quite a challenge to lecture in the traditional way and have students working independently or in groups. The lecture dynamics parallels the situation where one allows time for students to work independently on problems. Devitt (1990) and others have used this method.

(3) It is important to prepare for the time when students will have easy ***individual access*** to Symbolic Mathematical Systems. A consequence of technological improvements is that a calculator with integrated numeric, symbolic and graphic capabilities is no longer a dream and that such devices can only become progressively more powerful and cheaper. Furthermore, one can expect that the difference between portable computers and calculators will become less apparent. Denying the use of such calculators/computers in structured mathematics instruction does not solve the problem of their existence, and their access by a few more fortunate students. Every society believes that its students should be exposed to all environments which promise a richer educational experience. Of course many situations arise where that society cannot afford to provide a particular environment. Nevertheless this does not relieve teachers from their responsibility to make every possible effort to provide them.

c) Implications to the curriculum

There is no doubt that Symbolic Mathematical Systems will have impact on the ***curriculum***. What is in question is the magnitude of this impact. There is already evidence that traditional courses will have to change if these systems are to be integrated in any meaningful way. Even with relation to elementary concepts such as graphing, Dick and Musser (1990) observe: "This change in approach made possible by these calculators marks a significant shift in how graphing could be perceived by students. Instead of as a final task to be completed, graphing can assume the role as a problem-solving heuristic and a tool for exploration." Thus the traditional calculus approach of finding what the graph looks like is turned around to using calculus and numerical methods for locating more accurately the properties which are known to exist. Students rapidly come to appreciate both the exactness of non-numerical algebra and the approximation techniques underlying numerical analysis.

The decision as to what extent Symbolic Mathematical Systems are to be included in the mathematics curriculum will vary according to the groups of students being considered and their level. For instance, one could have requirements for a student in a university mathematics programme different from those for a student registered in a mathematics service course. In this respect there is much evidence that shows that scientists from other disciplines (see for example Lance et al. (1986)) serviced by mathematics departments are interested that their students not be denied the use of Symbolic Mathematical Systems. Such scientists, often more open-minded than pure mathematicians with respect to technological developments, simply perceive Symbolic Mathematical Systems as tools that can help them in their work and so are eager to use them. It is therefore necessary to reassess the proper balance in the requirement of basic symbolic manipulation skills and in the choice of topics covered in the various mathematics curricula.

Mathematics educators must make sure that in connection with domains where Symbolic Mathematical Systems can play a role, their courses help students acquire the appropriate intellectual skills. The required skills, while not really "new", are very often given little place in most traditional teaching: these are ***interpretive*** skills, needed to make mathematical judgements, to appreciate the validity and limitations of the tool being used, to assess the reasonableness of the computed "answer" (cf. Hodgson (1990)). Such skills, being much more demanding than traditional algorithmic ones, will require the student to be confronted with a substantial number of theoretical notions. Thorough understanding of mathematical concepts is thus now surely as — or even more — necessary in mathematics education as it has ever been (cf. Hodgson (1987)).

Another issue which is important in a pedagogical context is the extent to which the symbolic package will act in a "black box" mode or on the contrary give indications about how the "answer" to a particular problem can be obtained. A White-Box/Black-Box Principle has been advocated by Buchberger (1990) in relation to the question: Should students learn integration rules? Buchberger's point of view is essentially that in a stage where a certain mathematical topic is being learned by the student, the use of a Symbolic Mathematical System realizing the pertinent algorithms as "black boxes" would be a disaster. So he calls for systems that would feature the possibility to use an algorithm both as a "black box" (as is most often the case with existing systems) and as a "white box", i.e. in a step-by-step mode in which the reduction of the problem to subproblems is exhibited and in which the user could eventually interfere. A similar view is taken by Mascarello and Winkelmann (1992) in this volume. They claim that even if all the details of the internal functioning of a Symbolic Mathematical System are not usually essential to users, they must not remain totally hidden: understanding of main ideas and fundamental restrictions are necessary for proper use of (what they now call) the "grey" boxes.

Once these systems have been introduced into the mathematics courses then the student evaluation must change to reflect this new environment. As less emphasis is placed on certain techniques and more time is spent on concepts, the testing procedures must also change. Osborne (1990), Beckmann (1991) and others have started to address this issue.

It is clear that much experimentation and research are needed to establish how best to use Symbolic Mathematical Systems in the different courses, with the wide-ranging mathematical capabilities of students, and with the various attitudes of teachers. Appendix 3 provides a list of ongoing projects which are addressing some of these concerns.

3. Effecting curriculum changes

Generally curriculum changes in the secondary school system require much time to be implemented but when they happen, they are universally implemented: this is a direct consequence of the highly centralized administration of secondary school programmes in almost all educational systems. On the other hand curriculum changes in university courses can be far more spontaneous, but they tend to be localized to a particular course or section of a course, usually under the commitment of one or a few highly motivated individuals. Therefore the introduction of Symbolic Mathematical Systems in secondary school and university mathematics education poses problems of a different nature. In the former, to affect curriculum change one must convince a small group of influential curriculum makers. For the latter, to ensure that the use of Symbolic Mathematics Systems becomes integrated in courses, it is necessary to expose the majority of faculty members of the department to these systems. Kozma's (1985) study on instructional innovation in higher education supports this view. He contrasted projects which were collaboratively developed with those developed by individuals and found that the former were much more likely to be institutionalized. This section discusses some of the time and effort consuming activities which are required when introducing Symbolic Mathematical Systems in both upper secondary and university mathematics education.

The number of different Symbolic Mathematical Systems is expanding rapidly. Some of them have even been developed specifically for education at secondary school or at the beginning of university education. In most systems, especially the more recent ones, attention is being paid to make them more user friendly, that is, easier to use. A list of references which review some of the better known systems is provided in Appendix 1. While the choice of a specific Symbolic Mathematical System appropriate for use in a given classroom context might rest on various criteria (e.g. hardware facilities, level of instruction, topics to be covered, etc.), it is clear that some basic requirements must be met by those systems. For instance the use of the software should be transparent, that is students should spend their time thinking about the mathematics, and not how to operate the computer. Documentation should be essentially unnecessary for users, so that what needs to be done at any point should be apparent (some on-line "help" facility might however be useful in this respect). The software should be robust so that students' (sometimes unpredictable) behaviour should not cause it to crash or hang up too easily. It should interact easily with some word-processor, either internally to allow preparation of "notebooks" integrating word-processed text inserted in the middle of active symbolic software code, or externally to facilitate preparation of reports by students. But most important of all the program, whether used in a tutorial or interactive mode, should be devised so as not to foster the myth of computer omniscience and infallibility too often rooted in students' minds: while the computer brings in speed and reliability, it is the human being who has the intelligence and the ability to reason and make decisions.

As the cost of basic microcomputer technology continues to drop, one would hope for an analogous reduction in the price of hardware necessary for supporting Symbolic Mathematical Systems. While this has happened in some cases, this is not the general rule. Indeed, one should be aware that the general software development trend has been to demand more and more memory and disk space, thereby requiring more powerful and more expensive microcomputer units. Software developers tend to think in terms of the latest available (or forthcoming) hardware facilities, and experienced users call for more integration, namely word-processing, symbol and graphic manipulation, spreadsheets, etc., all of which push up the requirements of the computer system. Thus the implementation of curriculum change involving Symbolic Mathematical Systems requires financial planning for the purchase of equipment and software. Budgets must also be allocated to the maintenance of both hardware and software. Mathematics departments generally have little experience in requesting monies. This has tended to be the prerogative of Science, Physical Education, Fine Arts and other departments. In secondary schools funds are sometimes allocated for implementation of curriculum changes, but these are unlikely to be sufficient. In a university setting it may be worthwhile to run experimental sections to accumulate evidence of improvements in traditional indicators and to obtain faculty and student attitudes and responses to these systems.

In a context where a lot of importance is given in the literature to various symbolic software running on microcomputers, it might be tempting to overlook calculator technology. But the calculator is not restricted to school applications or to computations on numbers in so-called "scientific" notation! There are a number of calculator projects reported in the literature (see for example Nievergelt (1987) and Demana and Waits (1990)). It is true though that present calculators only have limited graphic and symbolic manipulation capabilities. But developments in electronic technology strongly suggest that such more powerful and user-friendly calculators will most certainly be a reality in a not too distant future. To equip a class or for individual use, calculator technology should thus be seriously considered. This is especially true in situations where, for instance, electricity supplies tend to be unreliable.

Once the equipment (hardware/software) has been purchased, meaningful mathematics activities for the students must be developed. Few such activities are available, although some recent publications provide examples in calculus: see for example the Mathematical Association of America Notes Series (P8) and the Maple Workbook (Geddes et al. (1988)) referenced in the Bibliography. But redefining objectives for a course or building pertinent activities is a daunting task. And for such a quest to have a lasting effect, it should be undertaken not by one individual (with eventual loss of the effect, should that individual be away for a while), but rather by a group, for instance by a majority of the faculty members within a mathematics department. This raises the difficult question of how to react to a possible lack of interest by some of those faculty members. After all, most are busy people and are not willing to invest large amounts of their limited time unless there is some evidence that the result will be worthwhile. This is even more true when students' attitudes towards the use of Symbolic Mathematical Systems in the classroom are not as positive as what could have been expected (see for instance Muller (1991) for an attitudinal survey of some teaching experience with a Symbolic Mathematical System).

The principal word of warning is certainly that implementing the necessary curriculum changes takes a lot of human resources in the form of time and dedication. It takes time to conceive the "new course", to develop meaningful students activities, to prepare new materials, to devise tools for assessment. And this must be done in contexts where often no (or little) credit is given to those who embark on such a task! Furthermore released time, supervision, hardware and software all require financial resources in an area where administrators have not been used to allocating funds. Mathematics educators must convince school or university administrations and funding bodies that such an investment is essential and is worth its value! And what is needed to support the argument is a critical analysis of controlled experiments, rather than anecdotal reporting of experiences.

4. Conclusion

The introduction of Symbolic Mathematical Systems into mathematics programmes should be considered within the broader context of the impact of technology on mathematics education. Mathematics teachers who have successfully integrated other software into their teaching of geometry, statistics etc. as well as computer scientists can offer useful insights and pedagogical points of view. Most of the projects aimed at the integration of Symbolic Mathematical Systems into mathematics teaching are either still under way or, if concluded, have results which are difficult to interpret. For example, how does one separate the effects of a Symbolic Mathematical System from other effects, such as those generated by the enthusiam of those involved with the experiment or the effects produced by the availability of additional resources? It is most probably too early to look for a significant impact on the curriculum (measured by the proportion of students in mathematics courses affected by the existence of Symbolic Mathematical Systems). It appears to be the consensus of those who are using these systems in their teaching that the course is taught differently but that it retains a fairly traditional content.

Thus there are few proposals of changes in the curriculum narrowly defined by course content. Some examples of proposals for change are: Tall (1985,1991) proposes a much greater visual component to calculus teaching; Möller (1990) suggests that the conceptual approach to calculus using

"Lipschitz-restricted" concepts of limit, continuity, differentiation and integration is a much more natural one for students and one in which Symbolic Mathematical Systems are easily integrated; Heid (1988) reports experiments in the resequencing of skills in introductory algebra and calculus where attention to hand manipulation skills was drastically reduced; Artigue et al. (1988) traces the influence of computers on the evolution of the teaching of differential equations; the texts of Hubbard and West (1991) and of Koçak (1989) support this evolution and emphasize the importance of visualization in the study of differential equations.

It is anticipated that many more such experiments will be reported in the near future as there are many projects on the way. Appendix 3 lists some of these projects for which information could be found. Ralston has constantly advocated curriculum reform at all levels of Mathematics Education in order to reflect the reality of today's technology and prepare individuals for future technology; in Ralston (1990), he proposes a framework for the school mathematics curriculum in 2000 which is highly dependent on the use of technology. Yet teachers receive their mathematics education from university mathematics courses in which they make very little use (if any!) of technology. How then can they be expected to realize the importance of technology in Mathematics Education? The reform must be spearheaded by the universities where there exists a greater latitude for experimentation.

There is as yet little evidence that Symbolic Mathematical Systems have had a significant impact on the mathematics curriculum of secondary schools and universities. It appears that the dominant reason for this lack of impetus on the curriculum is the education of teachers and faculty, that is, the lack of experience in these systems by a large proportion of mathematicians. In the university setting there is no evidence to suggest that changes implemented by an individual in one section of a course will have any impact on the course as a whole unless special effort is directed toward involving the majority of the faculty in a department. There are too many interests riding on the required introductory mathematics courses to expect that innovative changes made by one individual will be able to permeate the programme without the support from the majority of individuals in that department.

In spite of the human and financial costs involved, there is no doubt that Symbolic Mathematical Systems must be introduced into the mathematics curriculum. They probably constitute the single most powerful force compelling change in secondary and university mathematics education in the near future. They offer unprecedented opportunities to deepen and revitalize mathematics courses, focusing more on concepts and ideas than on mechanical calculations. While it is true that Symbolic Mathematical Systems, whether on microcomputers or on hand-held calculators, can only become more powerful, more user-friendly and more widely available, they offer right now an exceptional potential for progress in the teaching of mathematics and there is no reason for mathematics educators to delay becoming seriously involved with them. For such an evolution to happen, experiments must be performed on a very large scale and results must be evaluated and widely disseminated.

Appendix 3 contains a (partial) list of projects presently underway, in which Symbolic Mathematical Systems are being used in the classroom both at university and secondary school level. Hopefully these projects can stimulate more mathematics educators to involve Symbolic Mathematical Systems in their daily teaching.

Bibliography

Since the present volume updates the work and publications of the 1985 ICMI Study on "The Influence of Computers and Informatics on Mathematics and its Teaching" held in Strasbourg, this bibliography is restricted to references which have appeared since that meeting.

a) Proceedings

There are a number of conference proceedings, books of invited papers and series which provide an overview of classroom and/or laboratory projects and raise philosophical and cognitive issues of using a Symbolic Mathematical System in mathematics education. References P1 and P2 are the outcomes of the 1985 ICMI Study.

P1) The Influence of Computers and Informatics on Mathematics and its Teaching. Supporting Papers of the ICMI Symposium, IREM, Université Louis-Pasteur, Strasbourg, 1985.

P2) Howson, A.G. and Kahane, J.-P. (eds.), The Influence of Computers and Informatics on Mathematics and its Teaching. (Proceedings of the ICMI Symposium, Strasbourg, 1985). Cambridge University Press, 1986.

P3) Johnson, D.C. and Lovis, F. (eds.), Informatics and the Teaching of Mathematics. (Proceedings of the IFIP TC 3/WG 3.1 Working Conference, Sofia, 1987). North-Holland, 1987.

P4) Banchoff, T.F. et al. (eds.) Educational Computing in Mathematics. (Proceedings of ECM/87, Rome, 1987). North-Holland, 1988.

P5) Demana, F., Waits, B.K. and Harvey, J. (eds.) Proceedings of the Annual Conference on Technology in Collegiate Mathematics. (1st: 1988; 2nd: 1989; 3rd: 1990). Addison-Wesley, 1990, 1991.

P6) Cooney, T.J. and Hirsch, C.R. (eds.) Teaching and Learning Mathematics in the 1990s. (1990 Yearbook). National Council of Teachers of Mathematics, 1990.

P7) Dubinsky, E. and Fraser, R. (eds.) Computers and the Teaching of Mathematics: A World View. (Selected papers from ICME-6, Budapest, 1988). Shell Centre for Mathematical Education, University of Nottingham, 1990.

P8) The Mathematical Association of America has issued three volumes in the MAA Notes series related to this field and is preparing a fourth one: a) Smith, D.A. et al. (eds.), Computers and Mathematics: The Use of Computers in Undergraduate Instruction. MAA Notes Number 9, 1988. b) Tucker, T.W. (ed.), Priming the Calculus Pump: Innovations and Resources. MAA Notes Number 17, 1990. c) Leinbach, L.C. et al. (eds.), The Laboratory Approach to Teaching Calculus. MAA Notes Number 20, 1991. d) Computer Algebra Systems in Undergraduate Mathematics Education. To appear.

P9) The Notices of the American Mathematical Society feature a regular column under the title Computers and Mathematics (past editor: J. Barwise; current editor: K. Devlin).

b) Author Bibliography

In addition to the references mentioned in the text, the following list contains a selection of some useful papers or books.

Adickes, M.D., Rucker, R.H., Anderson, M.R. and Moor, W.C. [1991]: Structuring tutorials using Mathematica: Educational theory and practice, *Mathematica J.*, 1 (3), 86-91.

Akritas, A.G. [1989]: *Elements of Computer Algebra*, New York: John Wiley.

Artigue, M., Gautheron, V. and Sentenac, P. (1988]: *Qualitative study of differential equations: Results of some experiments with microcomputers* in Reference P4 above, 135-143.

Aspetsberger, K. and Kutzler, B. [1988]: *Symbolic computation — A new chance for education* in F. Lovis and E.D. Tagg (eds.) *Computers in Education*, 331-336, Amsterdam: North-Holland.

Aspetsberger, K. and Kutzler, B. [1989]: *Using a computer algebra system at an Austrian high school* in J.H. Collins et al. (eds.) *Proceedings of the Sixth International Conference on Technology and Education*, CEP Consultants Ltd., vol. 2, 476-479.

Auer, J.W. [1991]: *Maple Solutions Manual for Linear Algebra with Applications*, Englewood Cliffs, NJ: Prentice-Hall.

Ayers, T., Davis, G., Dubinsky, E. and Lewin, P. [1988]: Computer experiences in learning composition of functions, *J. for Res. in Math. Ed.*, 19, 246-259.

Beckmann, C.E. [1991]: *Appropriate exam questions for a technology-enhanced Calculus I course* in Reference P5 above (1989 Conference), 118-121.

Beilby, M., Bowman, A. and Bishop, P. [1991]: *Maths & Stats Guide to Software for Teaching* (2nd edition), CTI Centre for Mathematics and Statistics, University of Birmingham, UK.

Björk, L.-E. [1987]: *Mathematics and the new tools* in Reference P3 above, 109-115.

Bloom, L.M., Comber, G.A. and Cross, J.M. [1986]: Use of the microcomputer to teach the transformational approach to graphing functions, *Int. J. of Math. Ed.*, 17, 115-123.

Brown, D., Porta, H. and Uhl, J.J. [1990]: Calculus & Mathematica: Courseware for the Nineties, *Mathematica J.* 1 (1), 43-50.

Brown, D., Porta, H. and Uhl, J.J. [199-]: *Calculus & Mathematica*, Reading, MA: Addison-Wesley.

Buchberger, B. [1990]: Should students learn integration rules?, *SIGSAM Bull.*, 24, 10-17.

Capuzzo Dolcetta, I., Emmer, M., Falcone, M. and Finzi Vita, S. [1988]: *The laboratory of mathematics: Computers as an instrument for teaching calculus* in Reference P4 above, 175-186.

Cromer, T. [1988]: Linear algebra using muMATH, *Collegiate Microcomputer*, 6, 261-268.

Davenport, J.H., Siret, Y. and Tournier, E. [1988]: *Computer Algebra: Systems and Algorithms for Algebraic Computation*, Academic Press.

Dechamps, M. [1988]: *A European cooperation on the use of computers in mathematics*, in Reference P4 above, 197-209.

Demana, F. and Waits, B.K. [1990]: *Enhancing mathematics teaching and learning through technology* in Reference P6 above, 212-222.

Devitt, J.S. [1990]: *Adapting the Maple computer algebra system to the mathematics curriculum* in Reference P5 above (1988 Conference), 12-27.

Dick, T. and Musser, G.L. [1990]: *Symbolic/graphical calculators and their impact on secondary level mathematics* in Reference P7 above, 129-132.

Dubisch, R.J. [1990]: The tool kit: A notebook subclass, *Mathematica J.*, 1 (2), 55-64.

Ellis, W., Jr. and Lodi, E. [1989]: *Maple for the Calculus Student*, Pacific Grove, CA: Brooks/Cole.

Fey, J.T. [1989]: Technology and mathematics education: A survey of recent developments and important problems, *Educ. Studies in Math.*, 20, 237-272.

Flanders, H. [1988]: *Teaching calculus as a laboratory course* in Reference P4 above, 43-48.

Foster, K.R. and Bau, H.H. [1989]: Symbolic manipulation programs for the personal computer, *Science*, 243, 679-684.

Geddes, K.O., Marshman, B.J., McGee, I.J., Ponzo, P.J. and Char, B.W. [1988]: *Maple — Calculus Workbook*, University of Waterloo, Canada.

Gray, T.W. and Glynn, J. [1991]: *Exploring Mathematics with Mathematica*, Reading, MA: Addison-Wesley.

Heid, M.K. [1988]: Resequencing skills and concepts in applied calculus using the computer as a tool, *J. for Res. in Math. Ed.*, 19, 3-25.

Heid, M.K., Sheets, C. and Matras, M.A. [1990]: *Computer-enhanced Algebra: New roles and challenges for teachers and students* in Reference P6 above, 194-204.

Hodgson, B.R. [1987]: *Symbolic and numerical computation: The computer as a tool in mathematics* in Reference P3 above, 55-60.

Hodgson, B.R. [1990]: *Symbolic manipulation systems and the teaching of mathematics* in Reference P7 above, 59-61.

Hosack, J. [1988]: *Computer algebra systems* in Reference P8a above, 35-42.

Hubbard, J.H. and West, B.H. [1991]: *Differential Equations: A Dynamical Systems Approach, Part I: Ordinary Differential Equations*, New York: Springer-Verlag.

Hubbard, J.H. and West, B.H. [1991]: *MacMath: A Dynamical Systems Software Package*, New York: Springer-Verlag.

Koçak, H. [1989]:*Differential and Difference Equations through Computer Experiments* (2nd edition), New York: Springer-Verlag.

Kozma, R.B. [1985]: A grounded theory of instructional innovation in higher education, *J. of Higher Education*, 300-319.

Lance, R.H., Rand, R.H. and Moon, F.C. [1986]: Teaching engineering analysis using symbolic algebra and calculus, *Eng. Educ.*, 76, 97-101.

Mascarello, M. and Winkelmann, B. [1992]: *Calculus teaching and the computer. On the interplay of discrete numerical methods and calculus in the education of users of mathematics*, (in this volume).

Mathews, J.H. [1989]: Computer symbolic algebra applied to the convergence testing of infinite series, *Collegiate Microcomputer*, 7, 171-176.

Mathews, J.H. [1990]: Teaching Riemann sums using computer symbolic algebra systems, *College Math. J.*, 21, 51-55.

Möller, H. [1990]: *Elementary analysis with microcomputers* in Reference P7 above, 179-184.

Muller, E.R. [1991]: *Maple laboratory in a service calculus course* in Reference P8c above, 111-117.

Muller, E.R. [1992]: Symbolic mathematics and statistics software use in calculus and statistics education, *Zentralblatt Didaktik Math.* (to appear).

Neuwirth, E. [1987]: *The impact of computer algebra on the teaching of mathematics* in Reference P3 above, 49-53.

Nievergelt, Y. [1987]: The chip with the college education: the HP-28C, *Amer. Math. Monthly*, 94, 895-902.

Orzech, M. [1988]: *Using computers in teaching linear algebra* in Reference P8a above, 63-67.

Osborne, A. [1990]: *Testing, teaching and technology* in Reference P5 above (1988 Conference), 60-67.

Page, W. [1990]: Computer algebra systems: Issues and inquiries, *Computers Math. Applic.*, 19, 51-69.

Ralston, A. [1990]: *A framework for the school mathematics curriculum in 2000* in Reference P7 above, 157-163.

Shumway, R. [1990]: Supercalculators and the curriculum, *For the Learning of Math.*, 10 (2), 2-9.

Small, D. and Hosack, J. [1991]: *Explorations in Calculus with Computer Algebra Systems*, New York: McGraw-Hill.

Small, D., Hosack, J. and Lane, K. [1986]: Computer algebra systems in undergraduate Instruction, *Coll. Math. J.*, 17, 423-433.

Steen, L.A. [1988]: The science of patterns, *Science*, 240, 611-616.

Tall, D. [1985]: *Visualizing calculus concepts using a computer* in Reference P1 above, 291-295.

Tall, D. [1991]: *Recent developments in the use of computers to visualize and symbolize calculus concepts* in Reference P8c above, 15-25.

Wagon, S. [1991]: *Mathematica in Action*, San Francisco: Freeman.

White, J.E. [1988]: Teaching with CAL: A mathematics teaching and learning environment, *College Math. J.*, 19, 424-443.

White, J.E. [1989]: Mathematics teaching and learning environments come of age: Some new solutions to some old problems, *Collegiate Microcomputer*, 7, 203-224.

Young, G. [1986]: *Epilogue* in R.E. Ewing, K.I. Gross and C.F. Martin (eds.), *The Merging of Disciplines: New Directions in Pure, Applied and Computational Mathematics*, 213-214, New York: Springer-Verlag.

Zorn, P. [1987]: Computing in undergraduate mathematics, *Notices Amer. Math. Soc.*, 34, 917-923.

Zorn, P. [1990]: *Algebraic, graphical and numerical computing in elementary calculus: Report of a project at St. Olaf College* in Reference P5 above (1988 Conference), 92-95.

APPENDIX 1

This appendix provides a list of some Symbolic Mathematical Systems software reviews. It is important to realize that it is extremely difficult to evaluate and benchmark this software. Furthermore many of the evaluations do not take into account possible classroom use and the use by neophytes.

a) The *Notices of the American Mathematical Society* (see reference P9 above) have recently included an individual review of most Symbolic Mathematical Systems:

Vol. 35, 1988

The HP-28S brings computations and theory back together in the classroom, Y. Nievergelt, 799-804.

Supercalculators on the PC., B. Simon and R.M. Wilson, 978-1001.

Mathematica — A review, E.A. Herman, 1334-1344. (Also: *Other comments on Mathematica*, 1344-1349.)

Vol. 36, 1989

MicroCalc 4.0, G. Gripenberg, 680.

The menu with the college education (A review of Derive), E.L. Grinberg, 838-842.

Milo: The math processor for the Macintosh, R.F. Smith, 987-991.

Milo, Sha Xin Wei, 991-995.

PowerMath II, Y. Nagel, 1204-1206.

More on PowerMath II, P. Miles, 1206-1207.

Vol. 37, 1990

Review of PC-Macsyma, Y. Nagel, 11-14.

Review of True Basic, Inc. Calculus 3.0, J.R. Moschovakis, Y. Matsubara, G.B. White, 129-131.

Derive as a precalculus assistant, P. Miles, 275-276.

The right stuff, K. Devlin, 417-425.

Almost no stuff in, wrong stuff out, J.D. Child, 425-426.

Four computer mathematical environments, B. Simon, 861-868.

Vol. 38, 1991

Crimes and misdemeanors in the computer algebra trade, D.R. Stoutemyer, 778-785.

Periodic knots and Maple, C. Livingston, 785-788.

b) Other reviews are:

Symbolic manipulation programs for the personal computer, K.R. Foster and H.H. Bau, *Science*, 243, 679-684 (1989).

Derive: A mathematical assistant, E.A. Herman, *Amer. Math. Monthly*, 96, 948-958 (1989).

Mathematica: A system for doing mathematics by computer, L.S. Kroll, *Amer. Math. Monthly*, 96, 855-861 (1989).

Math without tears, C. Seiter, *MacWorld* 8 (1), 159-165 (1990).

Theorist, J. Rizzo, *MacUser*, 6 (6), 57-59 (1990).

Mathematica: A system for doing mathematics by computer, A. Hoenig, *Math. Intelligencer*, 12 (2), 69-74 (1990).

Theorist, F. Wattenberg, *Amer. Math. Monthly*, 98, 455-460 (1991).

Review of Maple in the teaching of calculus, E.R. Muller, *College Math. J.*, (to appear).

c) Reviews of software and comments on experiments on their use in teaching can also be found in specialized newsletters. Some examples are:

Computer Algebra Systems in Education Newsletter published by the Department of Mathematics, Colby College, Waterville, ME 04901, USA.

Maths & Stats published by the CTI Centre for Mathematics and Statistics (Computer in Teaching Initiative), Faculty of Education, University of Birmingham, Birmingham, B15 2TT, UK.

Computer-Algebra Rundbrief published by Fachgruppe 2.2.1 Computer-Algebra der GI, c/o Dr. F. Schwarz, GMD, Institut F1, Postfach 1240, 5205 St. Augustin, Germany.

APPENDIX 2

This appendix provides a limited number of examples to illustrate some of the capabilities of Symbolic Mathematical Systems (the system used here is Maple but this particular choice is not crucial). These systems are so powerful that it is impossible to provide a complete overview of their capabilities in a brief text.

#The system can be used to do some elementary
#number theory. For instance the command ifactor
#returns the prime factorization of an integer.

```
::: ifactor(123456780);
```

$$(2)^2 \quad (3)^2 \quad (5) \quad (47) \quad (14593)$$

#With such a tool available, it might be tempting to
#venture into some calculations that are not
#trivial to do either by hand or in a standard
#computer environment. For example the prime
#factors of the Mersenne number $2^{67}-1$ were
#given in 1903 by F. Cole. It reportedly took him
#"three years of Sundays" to complete the
#calculations. What can Maple do with that
#number?

```
::: ifactor(2^67-1);
```

$$(761838257287)(193707721)$$

#Done in just a fraction of a minute!! (But
#needless to say it is very easy to give as an
#input a number that would take "three years of
#Sundays" for the system to do.)
#Roots of equations can be found directly.

```
::: y:= x^3-4*x^2-7*x+10;
```

$$y := x^3 - 4\ x^2\ - 7\ x + 10$$

```
::: solve(y=0,x);
```

$$1, -2, 5$$

#Even with symbolic coefficients.

```
::: z:= a*x^2-2*b*x+c;
```

$$z := a\ x^2 - 2\ b\ x + c$$

```
::: solve(z=0,x);
```

$$1/2\ \frac{2\ b + 2\ (b^2\ - a\ c)^{1/2}}{a},$$

$$1/2\ \frac{2\ b - 2\ (b^2\ - a\ c)^{1/2}}{a}$$

#The example in Section 1 demonstrated that
#these systems are able to compute limits and to
#differentiate. They can also find Taylor's Series,
#sum finite and infinite series and integrate
#and solve differential equations.

```
::: taylor(ln(x^2),x=1,4);
```

$$2(x-1) - (x-1)^2 + 2/3\ (x-1)^3 + O((x-1)^4)$$

```
::: sum(j^2, j=1..m);
```

$$1/3\ (m+1)^3 - 1/2\ (m+1)^2 + 1/6\ m + 1/6$$

```
::: int(x^3*sqrt(x^2-9),x);
```

$$-3/5\ (x^2-9)^{1/2}x^2 - 54/5\ (x^2-9)^{1/2} + 1/5\ (x^2-9)^{1/2}x^4$$

```
::: deq:= diff(x(t),t)*t^3+x=0;
```

$$deq := (\frac{d}{dt}x(t))\ t^3\ + x = 0$$

```
::: dsolve(deq,x(t));
```

$$x(t) = \exp(1/2\ \frac{1}{t^2})C$$

#The next few examples are taken from linear
#algebra, namely, the solution of linear
#equations and some properties of matrices and
#vectors.

```
::: eqns:= a*x+b*y=e,c*x+d*y=f; vars:= x,y;
```

$$\mathbf{eqns} := \{a\ x + b\ y = e, c\ x + d\ y = f\}$$

$$\mathbf{vars} := \{x, y\}$$

```
::: solve(eqns,vars);
```

$$\{x = -\frac{b\ f - e\ d}{a\ d - c\ b}, \quad y = \frac{a\ f - c\ e}{a\ d - c\ b}\}$$

```
::: A:= array([[1,a,a^2],[1,b,b^2],[1,c,c^2]]);
```

$$\begin{aligned} \mathbf{A} := \mathbf{array}(&1..3, 1..3,\\ &[1, a, a^2]\\ &[1, b, b^2]\\ &[1, c, c^2]) \end{aligned}$$

```
::: det(A);
```

$$b\ c^2 - b^2 c - a\ c^2 + a^2 c + a\ b^2 - a^2 b$$

```
::: factor(");
```

$$-(-\ c + b)(a - c)(a - b)$$

```
#(The symbol " refers to the previously
#displayed expression.)
::: a1:= array([x1,y1,z1]); a2:= array([x2,y2,z2]);
  a3:= array([x3,y3,z3]);
```

$$\begin{aligned} \mathbf{a1} := \mathbf{array}(&1..3,\\ &[x1, y1, z1])\\ \mathbf{a2} := \mathbf{array}(&1..3,\\ &[x2, y2, z2])\\ \mathbf{a3} := \mathbf{array}(&1..3,\\ &[x3, y3, z3]) \end{aligned}$$

```
::: vol:= abs(dotprod(a1,crossprod(a2,a3)));
```

$$\begin{aligned} \mathbf{vol} := \mathbf{abs}(x1\ (&y2\ z3 - z2\ y3)\\ &+ y1\ (z2\ x3 - x2\ z3)\\ &+ z1\ (x2\ y3 - y2\ x3)) \end{aligned}$$

```
::: a:= array([[13,5],[5,2]]);
```

$$\begin{aligned} \mathbf{a} := \mathbf{array}(&1..2, 1..2,\\ &[13, 5]\\ &[5, 2]) \end{aligned}$$

```
::: c:= eigenvals(a);
```

$$c := 15/2 + 1/2\ 221^{1/2},\ 15/2 - 1/2\ 221^{1/2}$$

```
#The decimal approximation to these two
#eigenvalues gives
::: evalf(c[1]); evalf(c[2]);
```

$$14.93303438$$

$$.066965625$$

APPENDIX 3

There is as yet no single source which can provide a comprehensive international listing of projects in the area of Symbolic Mathematical Systems in Mathematics Education. Therefore, the following list cannot be regarded as comprehensive:

1: The Swedish ADM project (*Analysis of the role of the Computer in Mathematics Teaching*); see Björk (1987).

2: The Research Institute for Symbolic Computation at the Johannes Kepler University, Linz, Austria.

3: The Computers in Teaching Initiative Centre for Mathematics and Statistics (*Development of class work sheets to be used with Derive*), see the *Maths & Stats* newsletter published by the CTI Centre, University of Birmingham, UK.

4: *A European Cooperation on the use of Computers in Mathematics*; see Dechamps (1988).

5: The National Science Foundation (U.S.A.) is funding a number of different university projects specifically directed at integrating Symbolic Mathematical Systems into the calculus curriculum. The following is a selection providing a one line statement together with the university and the principal investigator.

Developing a user friendly interface to Maple and incorporating use of system into teaching calculus, Rollins College, Winter Park, FL; Douglas Child.

Developing new calculus curriculum using Maple on a VAX, Rensselear Polytechnic Institute, Troy, NY; William Boyce.

Developing a computerized tutor and computational aid based on Maple, University of Rhode Island, Kingston, RI; Edmund Lamagna.

Developing an electronically delivered course using the Notebooks feature of Mathematica, University of Illinois, Urbana, IL; Jerry Uhl.

Developing a new calculus course emphasing applications and using Mathematica, University of Iowa, Iowa City, IA; Keith Stroyan.

Developing a laboratory based calculus course using Mathematica, Iowa State University, Ames, IA; Elgin Johnston.

Developing a new calculus course for liberal arts colleges using Mathematica, Nazareth College, Rochester, NY; Ronald Jorgensen.

Developing calculus as a laboratory course using MathCad and Derive, Duke University, Durham, NC; David Smith.

Emphasizing computer graphics using Maple and emphasizing concepts via programming in ISETL, Purdue University, West Lafayette, IN; Ed Dubinsky.

Porting the laboratory calculus developed at Duke over to Mathematica, Bowdoin College, Brunswick, ME; William Barker.

Collecting, testing, and desktop publishing the best materials being developed using Mathematica, University of Michigan at Dearborn, Dearborn, MI; David James.

More detailed informations about projects in the U.S.A. integrating Symbolic Mathematical Systems in the calculus curriculum can be found in the reports contained in reference P8b above: Tucker, T.W. (ed.), *Priming the Calculus Pump: Innovations and Resources.* Mathematical Association of America (MAA Notes Number 17), 1990.

CALCULUS TEACHING AND THE COMPUTER. ON THE INTERPLAY OF DISCRETE NUMERICAL METHODS AND CALCULUS IN THE EDUCATION OF USERS OF MATHEMATICS

Maria Mascarello
Politecnico di Torino, I-10129 Torino, Italia

Bernard Winkelmann
Institut für Didaktik der Mathematik, D-4800 Bielefeld, Deutschland

1. NEW POSSIBILITIES

The computer is a mighty mathematical tool, not only for mathematical research, but even more in the process of applying mathematics and in the process of teaching and learning mathematics. In the following, we shall concentrate mainly on the new possibilities which the computer presents in the realm of calculus for users and future users of mathematics. By a user we mean somebody who is interested in mathematics merely (or mainly) through the use of mathematical models (in particular calculus models) to solve (extra-mathematical) problems. Future users of mathematics are, for example, engineering students, but even those learning calculus in schools as part of a general education may be included under this rubric.

1.1 New possibilities for the user

We describe first the changes in the mathematical knowledge and habits of the user of mathematics induced by the availability of sophisticated mathematical software to all who have to rely heavily on mathematical problem-solving such as engineers, natural scientists, etc. During the past decade we have seen the proliferation of mathematical software systems for personal computers which have become more powerful and/or more user friendly[1]. By raising the standards in these two domains, such systems are now in the hands of a rapidly growing number of users, even if until now (1991) they have not yet reached the majority of the teachers of mathematics, at least at the secondary level. But, if the trend continues, not only professional users of mathematics, but also most students and teachers will soon have regular access to such systems. However, the integration into regular classroom teaching will still be a problem.

The classic situation of the user of mathematics could be described - in a somewhat oversimplified manner - as a huge amount of passive mathematical knowledge contained in monographs, handbooks, recipes. Traditionally, this knowledge could only be used by being activated through the active mathematical knowledge of the user himself or by direct cooperation between the user and a mathematically more knowledgeable person. In contrast to this, the mathematical knowledge contained in mathematical software can have a far more active character, e.g. in giving advice and help interactively, offering possibilities for exploratory experiments or answering questions, acting like a mathematical expert system. Even more common numerical software, which exists in the form of sophisticated procedures, is far more active than the recipes of the old-fashioned handbooks, since in many cases these procedures are in fact polyalgorithms: They decide with considerable expertise which particular algorithm should be invoked, depending on the circumstances[2]. So the demand for mathematical knowledge on the part of the user has changed. The emphasis has shifted from detailed knowledge of the advantages and disadvantages of specific numerical methods and of the algorithms themselves to some meta-knowledge of the possibilities of numerical algorithms in general and their interaction with the concrete application situation.

As an example let us look at the process of the solution of ordinary differential equations[3]. This is indeed an example of great importance since such equations appear in many applications and are at the heart of applicable elementary calculus. So if it is possible to master them at a more elementary level than hitherto was possible, this could even be regarded as the most appropriate goal for the teaching of elementary calculus at schools and colleges. In the education of engineers at technical universities or similar institutions, where differential equations have always been part of the calculus sequence, even beginning calculus could concentrate more on applications and so give the student a more realistic and, one hopes, a more motivating start.

---

[1] A typical example might be the realm of computer algebra systems: In the progress from MUMATH to Derive there has been a big gain in user friendliness, allowing the use of the system even by users reluctant to program, but - at the same time - with a certain loss in functionality, e.g. in the solving of differential equations. On the other hand, the progress from MUMATH to Mathematica is mostly in power, much less in user friendliness. See also the chapter by Hodgson and Muller in this book.

[2] cf. Rice [1983], e.g. p.291f.

[3] cf. Winkelmann [1984] and the chapter by Tall and West in this book.

In the pre-computer age an engineer or scientist who had to handle differential equations was supposed to have detailed knowledge of diverse methods for the analytic solution of various elementary types, to be able to master complicated analytic-algebraic formulas and to carry out lengthy error-free symbolic and numerical calculations. Now he or she can use software which has this knowledge and ability built in, since it can solve more elementary differential equations than a non-specialist mathematician can do[4]. But in building up the model the user still has to understand fully the meaning and significance of the diverse quantities (variables) and of their derivatives and to be able to relate these to each other in order to set up the differential equation. And to give the details to the computer program, a thorough intuitive understanding of the mathematical meaning of the identifiers which appear in the modeling equations is needed, be it as variables, parameters, initial values, names for (yet unknown) functions (dependent variables) and so on. If an analytic solution exists, the program will normally present it as a somewhat confusing lengthy expression which must be qualitatively interpreted to be understood, namely through looking for simpler special cases, for settings of specific parameters or initial values, for asymptotic patterns of behaviour, etc. This process is guided by the intended interpretation of the solution in the context of the application model. If no analytic solution exists, the user may give his equation to some ready-made numerical software. In this case he needs some knowledge to make reasonable explorative choices of the values of parameters and initial values; there should be some experience with numerical phenomena (pitfalls of computations) and the ability to interpret the numerical and graphical output of the computer and to use this interpretation interactively for new choices of starting points for the next calculation.

In total, there can be observed a specific shift in the spectrum of abilities, from precise algorithmic abilities to more complex interpretations, so to speak from calculation to meaning, which in a certain sense is a reversal of the historical evolution. In this process the mathematics to be mastered tends to become intellectually more challenging, but technically simpler.

What does this mean for the mathematical education of the future user? Of course, there is no direct way from the mathematical activities of the user to the teaching process; the goal must not be confused with the means. Understanding and abilities for complex interpretations can only be built up by personal involvement of the student; she has to do full (but simpler) examples in all the main steps herself, be it by hand-calculating, by using interactive symbolic systems or calculators or by programming in some suitable programming language. This seems necessary in order to get an awareness of the mathematical situations, even if such activities are no longer part of the final application process. And even if today's sophisticated mathematical software need not and cannot generally be fully understood by the normal user, there must not be totally black boxes; a principal understanding of simple cases, of main ideas or of fundamental restrictions can be gained and seems necessary for proper use of the now 'grey' boxes[5].

On the other hand it is quite clear that extensive drill in formal calculations, in fluent structured programming or even in the handling of some software package cannot be justified in view of the changed qualifications needed by the user.

1.2 New possibilities in the teaching-learning process

In the field of teaching methods the computer, if it has been loaded with the appropriate software, will function as a simplifying aid, almost as a super hand-held calculator which permits the pupil to overcome computational obstacles in the treatment of more complex problems and to handle more realistic applications, e.g. in dealing with larger matrices, in the numerical solution of differential equations, or in the symbolic treatment of more complicated formulas; this will serve to widen the potential scope of mathematics education in terms of content. On the other hand, a computer equipped with appropriate languages and environments can become an instrument for solving problems in the hands of the student (interactive calculating or programming); in this case, the student tends to understand techniques more at the cognitive level, and no longer mainly at the level of skill. Beyond that, the computer, with its possibilities for illustration and symbolization, will provide opportunities for more comprehensive and rapid mathematical experiences.

---

[4] cf. Watanabe [1984].

[5] Buchberger [1990] gives an argument for a much more strict procedure: first, the algorithms of the software have to be completely understood by the student; afterwards he may use the software for all calculations. But Buchberger has the algorithms of Computer Algebra and mathematical majors or computer science majors in mind; his arguments do not extend to numerical software and typical future users of mathematics. See also the somewhat more detailed discussion in the chapter by Fraser, Klingen and Winkelmann in this book.

This presents problems and tasks as well as opportunities for educators mainly on two levels. On a more technical level, there is the necessity to provide more suitable software with strong mathematical functionality, educationally sound help functions, user interfaces for the inexperienced user, and accompanying explanations, hints and worked-out examples for teachers. On a more fundamental level, the problem is to achieve a balance between the quantitative and qualitative relation of new and old goals and methods as well as to set up the right trends for future developments.

The computer creates new opportunities for instruction in analysis, e.g.

- numerical and graphical illustrations[6],
- more complex and more realistic applications,
- a language in which to describe traditional calculus,
- CAL (computer-aided learning) in its various forms.

Some traditional motivations for treating conceptually exacting analysis in school can, however, no longer be maintained. For instance:

- calculations such as finding extreme values or areas can be easily done without analysis,
- practical applications in physics or technology which used to rely on analytic methods now are routinely done numerically on a computer by discrete calculations.

This results in a crisis: The legitimacy of traditional analysis in school is challenged; educators will have to make clear to the general public, and the teacher will have to explain to his pupils how and why the treatment of continuous analysis still makes sense nowadays.

In Section 3 we shall report on some experiments concerning the use of informatic tools in teaching basic mathematical courses at the Politecnico (Polytechnics) of Torino, Faculty of Engineering Sciences. We emphasize that the choice here has been to keep the teaching of calculus reasonably traditional, while at the same time giving some basic notions of informatics in the main course of lectures and devoting special laboratory sections to "calculus at the computer".

2. THE DISCRETE - CONTINUOUS INTERPLAY

2.1 General considerations

Although the role of applications of analysis has been changed both by the growing number of disciplines using mathematical models and by new methods, particularly the extensive use of computers, an understanding of fundamental concepts in which mathematizations take place remains indispensable. Examples are:

- variable quantity, change
- functional dependency
- local rate of change
- average value
- accumulation.

We shall refrain from discussing here how far traditional mathematics education was able to attain the goal of teaching these.

Now it is evident that these central concepts of mathematical applications can be implemented both by discrete and by continuous conceptualizations. Corresponding to such continuous concepts as function, differential equation, derivative, weighted integral, and integral, are the corresponding conceptualizations in discrete analysis, namely: sequence and time series, difference equation, difference, arithmetical mean value, and sum. These discrete concepts are often technically and almost always intellectually much simpler than their continuous counterparts.

In the following we will give some justifications, which are, in our opinion, crucial in answering the question now raised inevitably: "Why use the concepts of continuous analysis in teaching at all?" In (a) and (b), we state the problem, conceived as an epistemological question regarding the role of analysis in applications and model building, and in (c) and (d) we introduce the argument which solves the dilemma.

(a) *Insufficiency of continuous analysis for obtaining concrete numerical results.* Let us recall some of the facts: Most integrations cannot be executed analytically, but only numerically; this is all the more true for solving differential equations. Even tasks as simple as determining the extremes of a familiar function like $x \sin x$ require numerical methods. School mathematics has hitherto confined itself in a rather unnatural way to problems involving classes of functions which were solvable by analytic methods. It has paid dearly for this with heavy losses in orientation to problems of reality, content and relevance. This is particularly true for classical university courses in, for example, elementary differential equations[7].

---

[6] See Tall [1986], the article by D. Tall in Johnson/Lovis [1987] and the chapter by Tall and West in this book.

(b) ***Most concrete models using analysis have a discrete basis.*** This is first evident in the social sciences or in population biology, where the quantities to be modelled are numbers of items or individuals, or monetary units, which cannot be subdivided at will. But in physics, too, for instance, most models start discretely: even disregarding the fact that the universe is finite in principle and structured in particles, and that there are quanta (i.e. smallest units), it is a fact for quantities which are usually conceived of as being continuous, and mathematized accordingly, that concrete models based, say, on results of measurements, will start as discrete models simply because continuous functions cannot be obtained as the results of a series of measurements which yield only discrete sequences or time series. (This does not hold, of course, for modeling based on theoretical approaches.)

(c) ***The continuous character of models using analysis is the result of the intended domain of validity.***[8] Most mathematical models have a specific intended domain of validity, especially a certain scale level, even if this is not explicitly stated. A Euclidean line serves as a model for edges of solid bodies, e.g. of a shelf, only at a macroscopic scale. If we look at such an edge through an electron microscope, the edge doesn't look straight any more, and on the atomic scale, it loses its one-dimensional character too. Therefore, although the edge is well modelled by a line, we should not draw conclusions from this model outside its intended domain of validity. In an analogue sense, calculus models of discrete real phenomena typically are only intended for phenomena at scales where the discreteness doesn't enter. Calculus concepts such as limit, derivative, integral are not to be interpreted in the strict mathematical sense, but they express certain invariances: The corresponding discrete concepts do not depend on the step size, provided it is sufficiently small (but yet in the intended scaling domain). This consideration gives sense to the use of calculus models in such typically discrete domains as population dynamics or economics. But of course, there are also models, which do not show such invariances in their intended scaling domain. These should not be modelled by calculus. Such situations arise in considerations about fractal phenomena: the length of a coastline (as a quantity of integral type) is typically not invariant with the measuring unit, but of course the assumed statistical self-similarity also holds only in a sensible scale, which certainly does not extend to the microscopic level.

(d) ***The transition from models to concrete numerical results cannot be accomplished in general without continuous analysis.*** This is true, for one thing, because of the rounding errors which inevitably occur in numerical computing, and have to be controlled by a more abstract model which does not include the discretization error. A second, deeper reason follows from a closer look at the discrete aspects mentioned in points (a) and (b): It is the case that the step widths used in (a) and (b) are basically independent of each other, as is to be expected from the argument in (c). The density of the values measured in the measuring process is generally determined by practical considerations such as information content and "cost". One of the most fundamental hypotheses for determining the step width is that a diminution of the step width may yield more exact results, but basically not results which differ in principle. The phenomena which are to be observed and/or described are considered to be invariant with respect to the step width used in the observations provided it is sufficiently small. This fits in with the assumption that the corresponding limits exist. It is only on the basis of this assumption that the measuring process can be carried out in a discrete way chosen by practical considerations. In this case, however, the phenomena concerned are basically invariant with respect to the step width, and are thus best described in mathematical models which do not explicitly contain a step width. The fact that the step width with which the measured data were obtained is only of marginal importance for the model explains why step widths used, say, to solve numerically the corresponding differential equations, will generally be completely independent of the step width used in measurement. Both are independently determined by practical criteria such as cost and the precision required.

---

[7] This is properly described in Artigue [1989].

[8] We have taken this argument from Rice [1988] who writes under the sub-heading "Verifiable Hypotheses: Does Mathematics Model Reality?": "... we can argue that the real world is inherently discontinous everywhere, its 'microscopic' structure is either discrete or random or both. In any case, the mathematical definition of continuity, derivation, etc., do not apply because, at some fine scale of examination, the functions are undefined or discrete or something intractable. The implication of this view is that the concepts of smoothness and behaviors of functions are related to a scale and that an adequate mathematical model must take this into account." (p. 37).

This fundamental consideration has been reformulated here for the special case where the results of discrete measurement are used as a starting point. It is true, in an analogous way, pointed out in (c), for all the other cases in which mathematizing and modeling is done by analysis.

This behaviour is of course not valid for all mathematical models in the sciences or other domains. But it is in a sense typical for calculus models: If this behaviour is not observed in a specific situation, then normally we should really use discrete models, and if we - for technical reasons - nevertheless use some calculus models, we should be aware of the improper use and of possible difficulties in interpreting results. This may happen for example if we try to consider "fractal" phenomena in nature, such as natural borders (of islands, leaves of trees, etc.). Here, for example, the application of formulas for the length of a curve does not make much sense.

2.2 The context of dynamical systems

Dynamical systems (systems of time-independent explicit first order ordinary differential equations) appear as rather natural mathematical models for many situations in a variety of disciplines such as the physical, biological or economic sciences. Here typically we have to distinguish between situations where a natural step width exists whose value influences the phenomena, and situations in which this is not the case. In both cases, modeling with (discrete) difference equations is possible and adequate; but whereas in the former case, the step width of the difference equation has to be equal to that of the underlying situation, in the latter it may be chosen as a free parameter which suggests that the use of differential equations might be more natural.

As an example, consider the logistic growth of a (biological) population. If the generations of the population are distinct, as with certain bugs, there may be observed oscillations and fluctuations of the population, which are easily modelled and explained in the context of a difference equation, but would disappear in the transition to the corresponding differential equation (if it were not explicitly modelled by including a time lag which would induce similar fluctuations but would exclude the resulting equation from what is normally considered a differential equation in mathematics). But if generations are not distinct and population oscillations are slow compared to normal reproduction times, modeling with (logistic) differential equations seems adequate, even if there were only discrete points in time where new offspring could be noticed.

2.3 Symbolical, numerical and qualitative solutions

Linear differential equations, and some others which may be transformed to those, can be solved explicitly by closed formulas. From such formulas one can - at least in principle - answer almost any question about the underlying dynamical system: asymptotic behaviour, stability and dependence on initial values and parameters. But this is the exception, not the rule, since most dynamical systems arising from model building are essentially nonlinear[9] and do not admit any closed-form solutions. Numerical solution algorithms on the other hand are generally not sensitive to nonlinearity, but they share a double experimental character: in most cases, the degree to which they approximate the true solution can only be estimated, not proven[10]; and - more seriously - a numerical solution has a strict local empirical character. It does not by itself allow any conclusions about other initial values or parameters, which is catastrophic in applications where such values are only estimated. So they necessarily need to be complemented by theoretical, usually qualitative considerations about possible behaviours of this or a slightly modified dynamical system, be it continous or discrete. So this describes another complementarity between discrete numerical and theoretical methods.

3. EXPERIMENTS IN USING INFORMATIC TOOLS

In this section we report on some experiments concerning the use of informatic tools in teaching basic mathematical courses at the Politecnico of Torino (Italy), Faculty of Engineering Sciences. These experiments refer in particular to the courses Mathematical Analysis 1 and Mathematical Analysis 2 given to students of Mechanical Engineering in the years 1980 to 1983, using pocket computers. This activity was continued in 1984 and 1985, in the same courses, using such micro computers as the Sharp MZ803 and IBM PC. At this second stage, the experiment was concerned with a restricted number of students, selected on the basis of a test.

The experiment was sufficiently successful so that since 1986 all the students of the course (about 300) have been taught in the computer enhanced style. At the Politecnico of Torino an introductory computer laboratory is available for students; engineering students in the first two years have access to the lab after the completion of a specific course which prepares them for meaningful utilization of the available calculating devices and also supplies them with adequate knowledge of a programming language.

---

[9] For an interesting account of nonlinear model building see West [1985].

[10] An exception is the so-called EEE-methods, see Kaucher/Miranker [1984], but use of these methods is not yet widespread.

Several instructors of the Engineering Faculty have experimented with the use of the lab as an aid to the basic first two years mathematics courses, and from the resulting experience two didactic strategies have emerged. One was for the students themselves to perform the actual writing of the software. The other was to use existing software. It was observed that the preparation of software is, even from a mathematical standpoint, an occasion for investigation of the topic at hand. However, a certain risk was noted in the tendency toward interest in the computer itself to the detriment of time intended for dedication to mathematical reflection.

As far as concerns the use of already available software packages, the possibilities are many. We have readily available software written by colleagues instructing in analogous courses, that produced by students in previous courses, and, of course, software offered by the companies producing calculating devices.

While we refer to Boieri et al.[1984], to Mascarello-Scarafiotti [1987], [1988] and to Mascarello-Scarafiotti-Teppati [1989] for the general aims, the list of the themes and the results obtained, we should like to detail here some of the topics and content, and to add some final comments, as a 'proof' of what we asserted in Section 2.

Let us begin by observing that, to carry out the experiment in a useful way, it has been necessary to rely on basic informatic arguments. To this end, in the main course of lectures, the teacher, after giving some notions of the theory of formal languages, then introduced machine-numbers and algorithms for floating-point arithmetic computations. At the same time, in this first part of the course, some proofs of classical analysis results were presented in computational form.

One of the most important experiments concerned the study of dynamical systems using microcomputers. More specifically, we began in Mathematical Analysis 1 with the study of discrete dynamical systems, which was introduced after the study of sequences defined by recurrence formulas. As a natural continuation, in Mathematical Analysis 2 we considered continuous dynamical systems, giving a formal expression of the qualitative results. Finally, we returned to the use of microcomputers to find numerical results; this was done in order to check the known results of the theory, and also to conjecture new results concerning open problems.

To be specific, we briefly list the contents of the exercise sessions concerning dynamical systems (Mathematical Analysis 2):

- Cauchy problem for first order ordinary differential equations; solutions at the microcomputer, comparing the methods of Euler and Runge-Kutta.
- First order systems of ordinary differential equations, and in particular autonomous systems; visualization of the trajectories in the phase plane.
- Second order ordinary differential equations; solutions on the microcomputer of some nonlinear equations of particular significance in applications, such as the pendulum and other equations of mathematical physics.
- A numerical approach and simulation on the microcomputer of the trajectories for some problems which are still open in their qualitative aspects, as for example the mathematical model of the Lorenz attractor.

Now we present some further details of some of the above, which appear to us particularly significant from the didactic point of view. i) The student, knowing the classical analytic theory of linear equations with constant coefficients, and having some basic notions of the stability theory, is invited to "solve" the equation $\ddot{x} + k\dot{x} + x = 0$ on a microcomputer and to visualize the trajectories in the phase plane (without any direct assistance from the teacher). Figures 1 and 2 show some drawings of the kind obtained by a student.

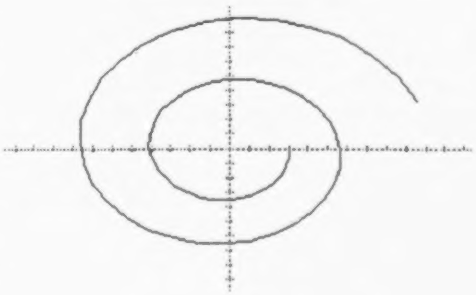


Figure 1: For the equation $\ddot{x} + x = 0, x(0) = 3, \dot{x}(0) = 0$ the Euler method converts what should be a circle to an outward spiraling curve.

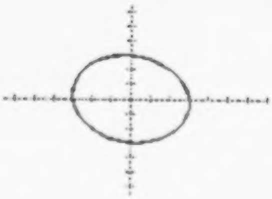


Figure 2: The true solution to $\ddot{x} + \dot{x} + 0.2x = 0, x(0) = 3, \dot{x}(0) = 0$ is an inwards spiraling curve. The outward spiraling which results with the Euler method exactly compensates resulting in no spiraling effect.

A discussion with the students followed concerning the validity of the results obtained in this way; particularly surprising is the second picture, where closed trajectories appear for $k \neq 0$. ii) The students "solve" on the microcomputer the pendulum equation $x'' + \sin x = 0$ by the Runge-Kutta method. Figure 3 shows the drawing obtained by one student.

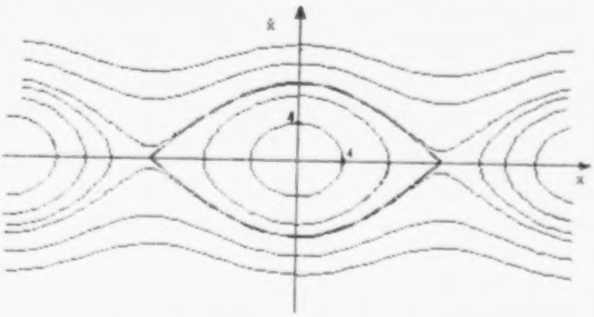


Figure 3: Phase portrait of the undamped pendulum, $\ddot{x} + \sin x = 0$ obtained by a student using a Runge-Kutta method.

We can observe that the picture seems satisfactory from a numerical point of view. Some qualitative aspects of the solutions are underlined by the teacher, as a check of the known results from the theory. iii) The student is invited to simulate on the screen the trajectories of the equation of the Lorenz attractor:

$$dx/dt = -sx + sy$$

$$dy/dt = rx - y - xz \quad \text{with} \quad s = 10, r = 28, b = 8/3$$

$$dz/dt = -bz + xy.$$

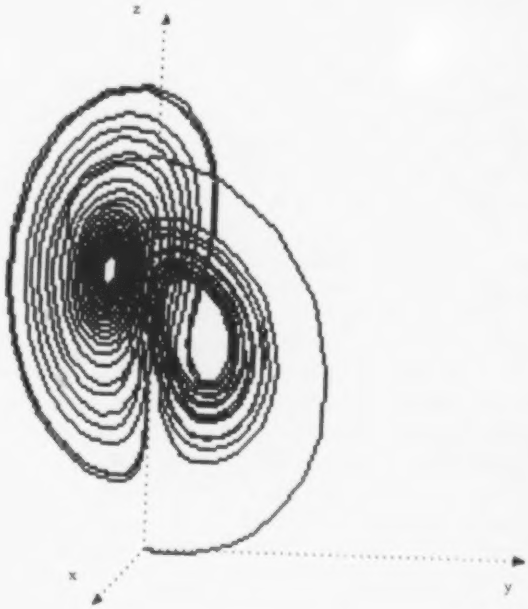


Figure 4: Plot of a curve approaching the Lorentz attractor as obtained by a student using the modified Euler method with $x(0) = 0.00001, y(0) = 0.00001$ and $z(0) = 0.00001$.

In Figure 4 there is a picture obtained by a student (the completion of the program required a certain informatic ability, due to the complications arising from the 3-dimensional representation of the trajectories in (x, y, z)-space). No comparison was attempted with known qualitative results since the existing literature on the subject seems to be too far advanced for a second year engineering student. However, a comparison was possible with what might be expected from the physical phenomenon (such as fluid turbulence phenomena). What it is very important to emphasize is that at this stage (end of Mathematical Analysis 2) students were able to evaluate correctly the results obtained from the computer, namely, to take into account the discrepancies which may occur between numerical solutions and analytic solutions, keeping in mind that the final objective is the interpretation of the physical phenomenon.

4. CONCLUSIONS

Our considerations have shown that even today when internally discrete digital computers are used for handling calculus models (so far as applications are concerned), continuous analysis cannot be dispensed with when describing problems for which analysis has been classically used. This, however, need not lead to the conclusion that analysis education at school or universities should go on as before. Our discussion has shown the function of continuous analysis in applications, and teaching must be done in such a way that this function is fulfilled. This requires that the transition from the discrete to the continuous model and vice versa be experienced by the students and that the respective particular possibilities and limitations of the model be perceived. To us, it would seem dishonest to try to explain to the student the importance of analysis for applications by means of unrealistic and oversimplified minimum-maximum tasks. Rather, it seems crucial to have the student at least begin to assess the usefulness of the various components of the system of analysis, i.e. concepts, approaches, calculi, translation schemes in practical applications. This goal should be attained by appropriate problem solving in the classroom; and formal explication should play a subordinate part. It remains to be seen how a balance between the individual components can be achieved. The following aspects, however, should be included in any case:

a) The teaching of analysis should include the treatment and study of discrete models. This leads to numerical computations. It does not necessarily imply explicit teaching of numerical mathematics, but requires including important basic numerical facts such as propagation of errors.

b) Building models is an important activity which must not be neglected in favour of just interpreting models. In particular, this means that the techniques of finding suitable functions are as important as discussing functions.

c) The role and function of (continuous) calculus must be developed in an appropriate way. It cannot be used to obtain numerical results, save in exceptional cases: it can, however, guide and direct the use of numerical methods.

d) The recent development of computer science has established techniques, in particular programming languages, which permit the precise description even of complicated processes such as, for instance, the algorithms necessary for symbolic differentiation. Mathematics teaching should increasingly make use of these results.

REFERENCES

Artigue, M. [1989]: Une recherche d'ingénierie di- dactique sur l'enseignement des équations différentielles en premier cycle universitaire in *Equipe de Didactique des Mathéma- tiques et de l'Informatique: Séminaire de Didactique des Mathimatiques et de l'Informatique*, Grenoble, Année 1988-1989. Université de Grenoble, LSD-IMAG, Institut Fourier.

Artigue, M. and Gautheron, V. [1983]: *Systèmes différentiels, Étude graphique*, Paris: CEDIC. ISBN 2-7124-0722-9.

Buchberger, B. [1990]: Should Students Learn Integration Rules?' *ACM SIGSAM Bulletin*, 24, 1, 10-17.

Boieri, P. et al. [1984]: Personal computers in teaching basic mathematical courses, *SEFI Annual Conference: The Impact of Information Technology on Engineering Education*, Erlangen.

Johnson, D. and Lovis, F. (Eds.) [1987]: *Informatics and the Teaching of Mathematics*, Proceedings of the IFIP TC 3/WG 3.1 Working Conference on Informatics and the Teaching of Mathematics, Sofia, Bulgaria, 16 - 18 May, 1987. Amsterdam: North-Holland. ISBN 0-444-70325-X.

Kaucher, E. and Miranker, W.L. [1984]: *Self-Validating Numerics for Function Space Problems*, San Diego: Academic Press.

Mascarello, M. and Scarafiotti, A.R. [1987]: Sperimentazione didattica nel Politecnico di Torino: Supporto informatica ai corsi die Analisi Matematica nel biennio della Facolta, *L'Educazione Matematica*, VIII, II, 2, 147-151

Mascarello, M. and Scarafiotti, A.R. [1988]: *Using computers in calculus examples-classes for engineers*, ECM/87 Educational Computing in Mathematics, Eds T.F. Banchoff et al, Amsterdam: North- Holland, 93-97.

Mascarello, M. and Scarafiotti, A.R. [1988]: *Experiments in mathematical education at secondary school and Politecnico of Torino (Italy)*, ICME 6, Theme Group 2 *Computers and the teaching of mathematics*, Working Group B.5 *The effects of technology and of computer science on a maths curriculum for the future*, preparatory papers, Budapest, 88-93.

Mascarello, M.; Scarafiotti, A.R.; and Teppati, G. [1989]: *Cultura e insegnamento: Esperienze significative appoggiate a metodi e strumenti informatici*, Proceedings of the congress /I Cultura Mathematica e Insegnamento/, Universita di Firenze, CDO, 253-260.

Rice, J.R. [1983]: *Numerical Methods, Software, and Analysis*, IMSL Reference Edition, New York: McGraw Hill.

Rice, J.R. [1988]: Mathematical Aspects of Scientific Software in J.R. Rice (Ed.), ***Mathematical Aspects of Scientific Software***, Springer, New York and in The IMA Volumes in ***Mathematics and Its Applications***, 14, 1 - 39.

Tall, D. [1986]: ***Building and Testing a Cognitive Approach to the Calculus Using Interactive Computer Graphics***, Ph.D. Thesis in Mathematics Education, The University of Warwick, Faculty of Education.

Watanabe, S. [1984]: An experiment toward a general quadrature for second order linear ordinary differential equations by symbolic computation in J. Fich (Ed.), *EUROSAM 84*, International Symposium on Symbolic and Algebraic Computation, Cambridge, England, July 9 - 11, 1984, Berlin: Springer-Verlag, 13 - 22.

West, B.J. [1985]: ***An Essay on the Importance of Being Nonlinear***, Lecture Notes in Biomathematics 62, Berlin: Springer-Verlag. ISBN 3-540- 16038-8.

Winkelmann, B. [1984]: The Impact of the Computer on the Teaching of Analysis, *Int. J. Math. Educ. Sci. Tech.*, 15, 675 - 689.

Winkelmann, B. [1989]: ***Dynamische Systeme und Differentialgleichungen. Einige mathematische Anmerkungen zu dynamischen Systemen und ihrer Simulation***, LOG IN 9, Heft 4, 19 - 23. ISSN 0720-8642.

GRAPHIC INSIGHT INTO MATHEMATICAL CONCEPTS

David Tall
Mathematics Education Research Centre
Warwick University, Coventry, U.K.

Beverly West
Mathematics Department
Cornell University, Ithaca, NY 14583, U.S.A.

The human brain is powerfully equipped to process visual information. By using computer graphics it is possible to tap this power to help students gain a greater understanding of many mathematical concepts. Furthermore, ***dynamic*** representations of mathematical processes furnish a degree of psychological reality that enables the mind to manipulate them in a far more fruitful way than could ever be achieved starting from a static text and pictures in a book or roughly drawn pictures on a chalk board or overhead projector. Add to this the possibility of student exploration using prepared software and the sum total is a potent new force in the mathematics curriculum.

In this paper we report on the development of interactive high resolution graphics approaches to various areas in mathematics. The first author has concentrated initially on the calculus in the UK (Tall, 1986, Tall et al, 1990) and the second is working in the USA on differential equations with John H. Hubbard (Hubbard and West, 1990).

An interactive visual approach is proving successful in other areas, for example, in geometry (*The Geometric Supposer, Cabri Géomètre*), in data manipulation (e.g. *Macspin, Mouse Plotter*), in probability and statistics (e.g. Robinson and Bowman, 1987) and, more generally, in a wide variety of topics (such as the publications in the *Computer Illustrated Text* series, which use computer programs to provide dynamic illustrations of mathematical concepts).

New approaches to mathematics

The existence of interactive visual software leads to the possibility of an exploratory approach to mathematics which enables the user to gain intuitive insight into concepts, providing a cognitive foundation on which meaningful mathematical theories can be built. For example, the notion of a limit has traditionally caused students problems (e.g. Cornu, 1981, Tall and Vinner, 1981). The computer brings new possibilities to the fore; we may begin by considering the gradient not of the tangent, or of a chord as it approaches a tangential position, but simply ***the gradient (or slope) of the graph itself.*** Although a graph may be curved, under high magnification a small part may well look almost straight. In such a case we may speak of the gradient of the graph as being the gradient of this magnified (approximately straight) portion. For instance, a tiny part of the graph $y = x^2$ near $x = 1$ magnifies to a line segment of gradient 2 (figure 1).

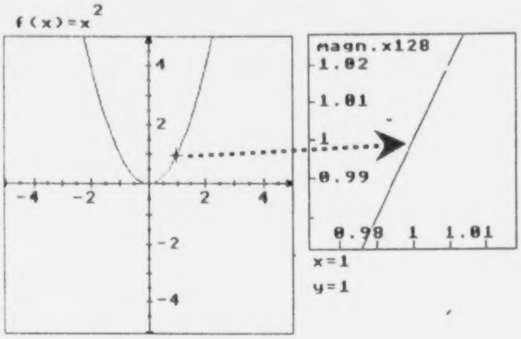


Figure 1: Magnifying a small part of a graph to show its local straightness.

To represent the changing gradient of a graph, it is a simple matter to calculate the expression $(f(x+c) - f(x))/c$ for a small fixed value of c as x varies. As the chord clicks along the graph for increasing values of x, the numerical value of the gradient for each successive chord can be plotted as a point and the points outline the graph of the gradient function (figure 2). In this case the chord gradient function of $\sin x$ for small c approximates to $\cos x$, which may be checked by superimposing the graph of the latter for comparison. Thus the gradient of the graph may be investigated experimentally before any of the traditional formalities of limiting processes are introduced.

Such moving graphics also enable the student to get a dynamic idea of the changing gradient. Students following this approach can see the gradient as a global *function*, not simply something calculated at each individual point.

The symbols dx, dy can also be given a meaning as the increments in x, y to the tangent. Better still, (dx, dy) may be viewed as the *tangent vector*, a valuable idea when we come to the meaning of differential equations.

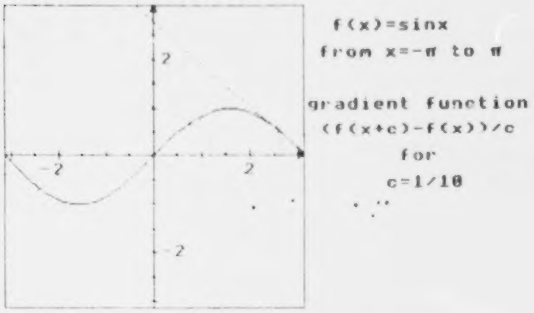


Figure 2: Building up the gradient function of a graph.

Conceptualizing non-differentiable functions

In a traditional calculus course, non-differentiable functions would not be considered until a very late stage, if at all. However, if one views a differentiable function as one which is "locally straight", then a non-differentiable function is simply one which is *not* locally straight. For instance, the graph of $|x-1|$ at $x=1$, or $|\sin x|$ at $x=\pi$, has a "corner" at the point concerned with different gradients to the left and right. More generally, it is possible to draw a function that is so wrinkled that it never looks straight *anywhere* under high magnification.

An example is the *blancmange function* $bl(x)$, first constructed by Takagi in 1903. First a sawtooth $s(x)$ is constructed for a real number x by taking its decimal part $d = x - INT(x)$ and defining

$$s(x) = \begin{cases} d & \text{if } d < \frac{1}{2} \\ 1-d & \text{otherwise.} \end{cases}$$

The sequence of functions

$$\begin{aligned} b_1(x) &= s(x) \\ b_2(x) &= s(x) + s(2x)/2 \\ &\vdots \\ b_n(x) &= s(x) + \ldots + s(2^{n-1}x)/2^{n-1} \\ &\vdots \end{aligned}$$

tends to the blancmange function (figure 3).

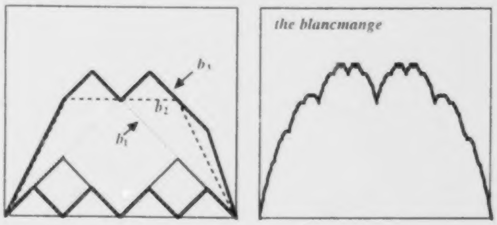


Figure 3: Building up the blancmange function adding successive half-size sawtooth graphs.

The process may be shown dynamically on a visual display unit; we regret that it cannot be pictured satisfactorily in a book. But higher magnification of the blancmange function using prepared software shows it can nowhere be magnified to look straight, so it is nowhere differentiable. This intuitive approach can easily be transformed into a formal proof of disarming simplicity (Tall 1982).

Visualizing solutions of first order differential equations

In graphical terms, a first order differential equation $dy/dx = f(x,y)$ simply states the gradient of a solution curve at any point (x,y) and a solution is simply a curve which has the required gradient everywhere. The *Solution Sketcher* (Tall 1991) or *MacMath* (Hubbard and West 1991) allows the user to point at any position in the plane and draws a small line segment of the appropriate direction. This line-segment may be marked on-screen and successive line segments fitted together to build up an approximate solution curve. More broadly, it is possible to draw a direction diagram with an array of such segments and to trace a solution by following the given directions (figure 4).

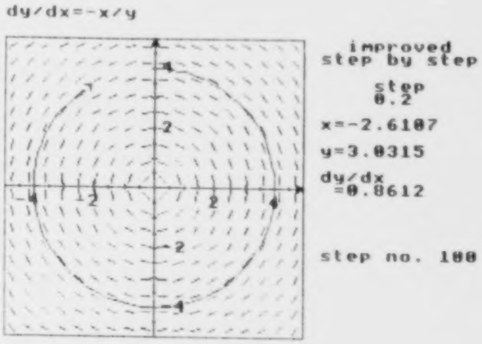


Figure 4: Drawing a numerical solution of a first order differential equation.

The differential equation

$$y\frac{dy}{dx} = -x$$

has implicit solutions of the form $x^2+y^2 = k$, rather than an explicit global solution of the form $y = f(x)$. At points where the flow-lines meet the x-axis, the tangents are vertical and the interpretation of dy/dx as a function fails, but the vector direction (dx, dy) is valid with $dx = 0$ and $dy \neq 0$. Thus a first-order differential equation is sometimes better viewed in terms of the direction of the tangent to a solution curve rather than specifying the derivative.

Existence of solutions

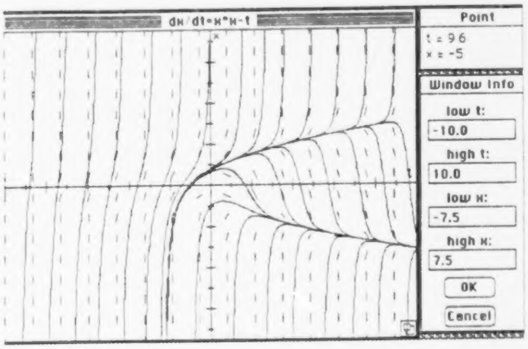
There comes a time in every university course on differential equations when honesty should compel the teacher to admit that cookbook methods for solving differential equations are inadequate. Such innocent looking equations as

$$dy/dx = y^2 - x, dy/dx = \sin(xy), dy/dx = e^{xy}$$

do not have solutions that can be written in terms of elementary functions. Students often mistakenly confuse this with the idea that the equations have no solutions at all. However, if they are able to interact with a computer program that plots a direction field and then draws solutions numerically following the direction lines, the notion of a solution takes on a genuine meaning: "Of course the equations have solutions: we can *see* them!" From this cognitive base it is possible to use the computer to analyse solutions in an entirely new way.

Qualitative analysis of differential equations

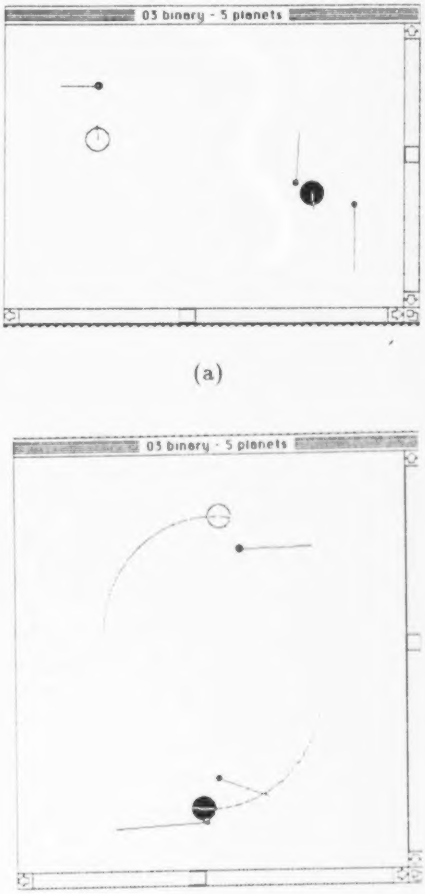
New forms of analysis emerge now that we can see as many solutions as we wish all at the same time. In figure 5, notice how the solutions tend to "funnel" together moving to the lower right-hand side; in the upper right they spray apart (an "antifunnel"). Qualitatively descriptive terms such as "funnel" and "antifunnel" can be defined precisely to give powerful theorems with accurate quantitative results (Hubbard and West 1991). For example, the equation $dy/dt = y^2 - t$ in figure 5 has two overall behaviours: solutions either approach vertical asymptotes for finite t or fall into the funnel and approach $y = -\sqrt{t}$ as $t \rightarrow +\infty$. In the antifunnel there is a unique solution approaching $y = +\sqrt{t}$ which separates the two usual behaviours. Furthermore, the qualitative techniques enable us to estimate the vertical asymptote for a solution through any given point with any desired precision.



t = 96
x = -5

low t: -10.0
high t: 10.0
low x: -7.5
high x: 7.5

Figure 5: A family of solutions of a differential equation, showing funnel and antifunnel behaviour.



(a)

(b)

Figure 6: A numerical approximation to the many-body problem. (a) Masses in initial position with velocity vectors. (b) A little later under the action of Newton's Laws.

Newton's Laws

The classical three body problem defies elementary analysis, yet a computer program can cope with relative ease. The program *Planets* (Hubbard and West 1990) takes a configuration of up to ten bodies with specified mass, initial position and velocity and displays the movement under Newton's laws (figure 6). The data can be input either graphically with the cursor, or numerically in a table. The program allows exploration of possible planetary configurations and it soon becomes plain that stability is the exception rather than the rule. One may wonder under what circumstances stability occurs. Other questions arise, such as the reason for the braided rings of Saturn that were a great surprise when first observed by the Voyager space flight. Nobody had imagined such a behaviour beforehand, yet braided behaviour showed up in the very first experiments with the *Planets* program.

Figure 7 shows a model of a possible orbit of a tiny satellite around two larger bodies, alternately oscillating between revolving round one then moving into a position of superior gravitational pull of the other and moving, for a time, to revolve round the other (Koçak 1986). Once again, computer exploration shows vividly how three bodies move in a complex pattern.

The theory of dynamical systems and chaos is a paradigmatic example of a new branch of mathematics in which the complementary roles of computer-generated experiments to suggest theorems and formal mathematical proofs to establish them with logical precision go hand in hand.

> Chaos has become not just a theory but also a method, not just a canon of beliefs but also a way of doing science. ... To chaos researchers, mathematics has become an experimental science, with the computer replacing laboratories full of test tubes and microscopes. Graphic images are the key. "It's masochism for a mathematician to do without pictures" one chaos specialist would say. "How can they see the relationship between that motion and this, how can they develop intuition?". (Gleick 1987, pp. 38-39)

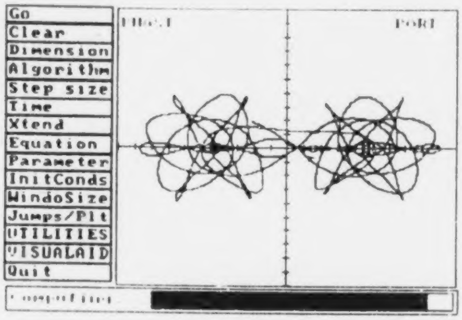
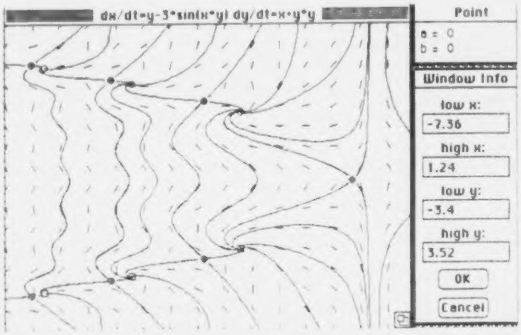


Figure 7: A numerical plot representing a tiny satellite orbiting two larger bodies.

Systems of differential equations

The *MacMath* software of Hubbard and West (1991) draws solutions of systems of differential equations $dx/dt = f(x,y), dy/dt = g(x,y)$ in the x,y- plane and also locates singular points using Newton's method, drawing separatrices for saddle points (figure 8).


dx/dt=y-3*sin(x*y) dy/dt=x+y*y

Point

a = 0
b = 0

Window Info

low x: -7.36
high x: 1.24
low y: -3.4
high y: 3.52

Figure 8: Locating singular points and separatrices for saddle points.

In this way the computer may be used to draw solutions of systems of differential equations that are far too complicated to draw by hand. As a further example, Artigue and Gautheron (1983) draw the solutions of the polar differential equations

$$\frac{dr}{dt} = \sin r, \frac{d\theta}{dt} = \cos r$$

which exhibit limit cycles for $r = k\pi$ (figure 9).

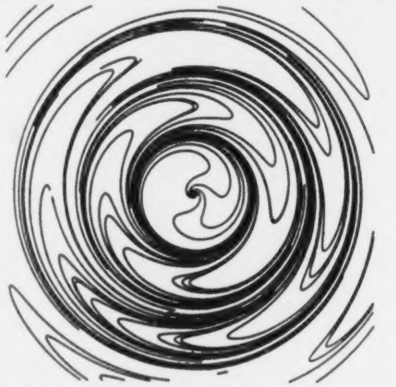


Figure 9: Limit cycles of simultaneous polar differential equations.

Generalizing the concept of visual solutions

A second order differential equation such as

$$\frac{d^2x}{dt^2} = -t$$

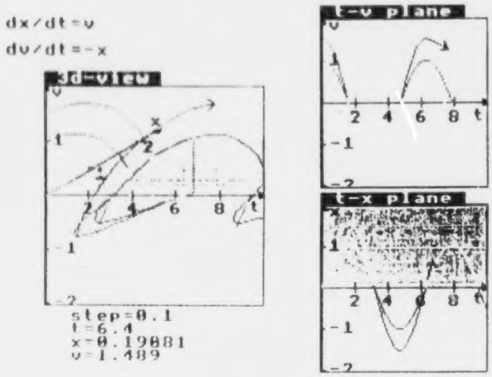
no longer has a simple direction field in (t,x) space, because through each point (t,x) there is a different solution for each starting direction $v = dx/dt$. However, this differential equation is equivalent to the simultaneous linear equations:

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = -t$$

and in three dimensions, with coordinates (t,x,v), these equations determine a unique tangent vector (dt, dx, dv) in the direction $(1, v, -t)$. Hence the idea of a direction field does generalize, but it must be visualized in three-dimensional (t,x,v) space. Figure 10 shows two solutions of the simultaneous differential equation spiralling through (t,x,v) space and their projections onto the $t-x$ and $t-v$ planes, with the $t-x$ projection giving solutions to the original second order differential equation.

Visual exploration in geometry

Euclidean geometry traditionally served to introduce students to a deductive system. In many countries (such as the United Kingdom) it has all but disappeared from the mathematics curriculum. Computers now give the opportunity to manipulate geometrical figures to build up intuitions for possible theorems (the *Geometric Supposer*, Schwartz and Yerushalmy, 1985, *Cabri Géomètre*, 1987). The initial phase of study of geometry can now be an experimental science, in which the student can use the computer to construct a figure and experiment with it.


3d-view
v
1
-1
-2
2
4
6
8
t
x
2
t-v plane
v
1
2
4
6
8
t
-1
-2
t-x plane
x
1
2
4
6
8
t
-1
-2

```
dx/dt=v
dv/dt=-x
step=0.1
t=6.4
x=0.19881
v=1.489
```

Figure 10: Two nearby solution curves for a pair of simultaneous differential equations.

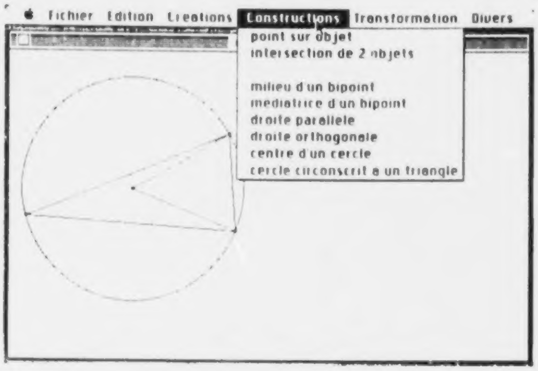

Fichier
Edition
Creations
Constructions
Transformation
Divers
point sur objet
intersection de 2 objets
milieu d'un bipoint
mediatrice d'un bipoint
droite parallele
droite orthogonale
centre d'un cercle
cercle circonscrit a un triangle

Figure 11: Cabri Géomètre software for manipulating geometric figures.

Visual Data Processing

It is now possible to explore data visually, for example, to see a line of best fit for data in two or three dimensions. *MacSpin* allows up to ten categories of data, from which any three can be selected and displayed. Though only represented as a projection of three dimensions onto the two-dimensional screen, the data may be rotated and viewed dynamically from any angle to give a sense of depth that is not visible in a static picture (figure 12). Individual points may be selected and inspected to see where the data originates to identify interesting information, such as outlying values. Rotating the data in the figure suggests that it clusters together in a way which intimates that the three components are correlated.

Modern spreadsheets, statistical packages and data handling packages now include visual representation of data which encourages the user to explore and communicate complex information in visual ways.

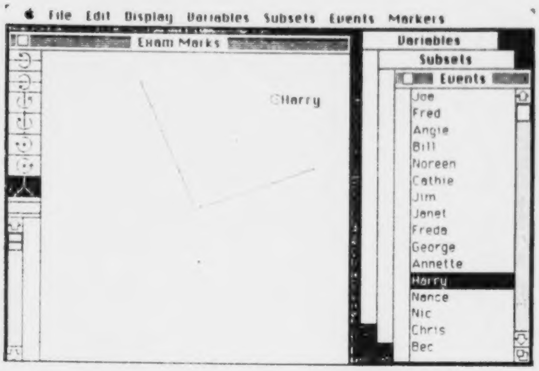


Figure 12: Manipulating data with three components to look for a visual correlation.

The ability to present and manipulate information visually is becoming widely available in many different areas in mathematics. For example, Robinson and Bowman (1986) introduce probability and statistics using computer graphics with the intention of giving a 'feel' for probability distributions rather than elaborating mathematical detail. More generally, the *Computer Illustrated Texts* (starting with Harding 1985) are designed to use simple computer programs to provide interactive illustrations of mathematical ideas which can be explored by the student in place of static pictures in a book.

Is programming essential?

We have not explicitly mentioned programming for the purpose of gaining insight into mathematical processes. A body of expertise is growing in which students are expected to write or adapt short programs (usually in structured Basic, Pascal, or Logo) to carry out mathematical algorithms. From here it is often intended that they move on to prepared software that uses the underlying algorithms in a more interactive manner. The early computer-illustrated texts assumed that the programming would be sufficiently simple that it would allow the student to modify the programs, but this became an impossible ideal in later texts as more sophisticated programs were written that were too complex for the user to modify. Programming requires a serious investment in time and effort. However, it can pay vast dividends in gaining insight into the underlying mathematical processes if the investment is sufficiently generous.

Dubinsky has evidence that having students make certain programming constructions (in the computer language ISETL) can lead to their making parallel mathematical constructions in their minds and thereby come to understand various mathematical concepts (see, for example, Dubinsky and Schwingendorf 1991). Clearly a spectrum of approaches may be possible with varying amounts of programming, depending on the time and commitment available.

New Styles of Learning

Software is becoming widely available to give graphical representations in calculus, differential equations, geometry, data handling, numerical analysis, and many other areas of mathematics. This is usually predicated on a new kind of learning experience Q one in which the student may *explore and manipulate* ideas, *investigate* patterns, *conjecture* theorems and *test* theories experimentally before going on to *prove* them in a more formal context.

For instance, beginning calculus students may investigate the gradients of functions such as sine, cosine, tangent, exponential and logarithm, and conjecture their formulas before they are derived formally (Tall 1986, 1987). In differential equations they may explore problems at the boundaries of research (such as the rings of Saturn) and make the mental link between the friendly world of (mostly linear) equations that can be solved by formulas and the strange world of those (usually nonlinear) that can not (Hubbard and West 1991).

This form of learning is not a *replacement* for formal deduction, but a *precursor* and a *complement* to it. It enables the less able student to grasp essential ideas that would previously be too difficult when framed in a purely formal theory and for the more able student to build a cognitive base for the formal theory to follow. It enables a wide range of students to integrate their knowledge structure through their powers of visualization.

Acknowledgement

The authors are grateful to Professor John H. Hubbard for his assistance in the preparation of an earlier version of this article.

REFERENCES

Alfors, D. and West, B. [1992]: *Analyzer* calculus software* (for the Macintosh computer), Reading, MA: Addison-Wesley.

Artigue, M. and Gautheron, V. [1983]: *Systèmes Différentiels: Étude Graphique*, CEDIC, Paris.

Bach, J.O. [1990]: *MultiMat* (for I.B.M. compatible computers; available in English), Copenhagen: Danish Mathematics Teachers Association.

Cabri Géomètre [1987]: IMAG, BP 53X, Université de Grenoble (for IBM and Macintosh computers).

Cornu, B. [1981]: Apprentissage de la notion de limite : modèles spontanés et modèles propres, *Actes du Cinquième Colloque du Groupe International P.M.E.*, Grenoble, 322-326.

Devaney, R.L. [1990]: *Chaos, Fractals, and Dynamics: Computer Experiments in Mathematics*, Reading, MA: Addison-Wesley.

Dubinsky, E. and Schwingendorf, K. E. [1991]: *Constructing calculus concepts: cooperation in a computer laboratory* in *The Laboratory Approach to Teaching Calculus* (Leinbach, L.C., Ed.), MAA Notes Series No. 20, Washington, DC: Mathematical Association of America.

Gleick, J. [1987]: *Chaos: Making a New Science*, London: Penguin and Viking: New York.

Harding, R. [1985]: *Fourier Series and Transforms, A Computer Illustrated Text*, Bristol and Boston: Adam Hilger (for the BBC, IBM and Apple Computers).

Hubbard, J.H. and Parmet, M. [1987]: *3D Analyzer; Complex Paint* (for the Macintosh computer), Ithaca, NY: Cornell University.

Hubbard, J.H. and West, B. [1991]: *Differential Equations: A Dynamical Systems Approach*, New York: Springer-Verlag.

Hubbard, J.H. and West, B. [1991]: *MacMath: A Dynamics Package* (for the Macintosh computer), New York: Springer-Verlag.

Johnson, J.A. [1990]: *Gyrographics* (for I.B.M. compatible computers), Stillwater, OK: Cipher Systems.

Klotz, E. and Jackiw, N. [1991]: *The Geometer's Sketchpad* (for the Macintosh computer), Berkeley, CA: Key Curriculum Press.

Koçak, H. [1986]: *Phaser: Differential and Difference Equations through Computer Experiments* (for IBM computers), New York: Springer-Verlag.

MacSpin [1985]: D^2 Software Inc., Austin, TX (for the Macintosh Computer).

Phillips, R. [1988]: *Mouse Plotter*, Shell Centre, Nottingham (for the Archimedes computer).

Robinson, D.A. and Bowman, A.W. [1986]: *Introduction to Probability*, Bristol and Boston: Adam Hilger (for the BBC and IBM computers).

Schwartz, J. and Yerushalmy, M. [1985]: *The Geometric Supposer*, Sunburst Communications, Pleasantville, N.Y. (for the Apple Computer).

Takagi, T. [1903]: A simple example of a continuous function without derivative, *Proc. Phys.-Math.*, Japan, 1, 176-177.

Tall, D.O. [1982]: The blancmange function, continuous everywhere but differentiable nowhere, *Mathematical Gazette*, 66, 11-22.

Tall, D.O. [1986]: *Graphic Calculus I-III* (for BBC compatible computers), Glentop Press, London.

Tall, D.O. [1987]: *Readings in Mathematical Education: Understanding the Calculus*, (collected articles from *Mathematics Teaching*, 1985-7), Association of Teachers of Mathematics, UK.

Tall, D.O. [1991]: *Real Functions and Graphs* (for BBC compatible computers), Cambridge: Cambridge University Press.

Tall, D.O., Blokland, P. and Kok, D. [1990]: *A Graphic Approach to the Calculus* (for I.B.M. compatible computers), Sunburst Inc., USA.

Tall, D.O. and Vinner, S. [1981]: Concept image and concept definition in mathematics, with special reference to limits and continuity, *Educational Studies in Mathematics*, 12 151-169.

Zimmerman, W. and Cunningham, D. [1990]: *Visualization in Teaching and Learning Mathematics*, MAA Notes Number 19, Washington,DC: Mathematical Association of America.

Annotated References

This is a list of some of the references earlier in this book of particular significance or usefulness with a brief annotation describing the contents of the article or book.

Aho, A.V., Hopcroft, J.E. and Ullman, J.D. [1983]: *Data Structures and Algorithms*, Reading, MA: Addison-Wesley.

This is a graduate text that presents the data structures and algorithms that underpin much of today's computer programming. It covers these topics in the context of solving problems using computers and introduces step counting and complexity as an integral part of problem solving. It is very nicely written and comprehensive.

Alfors, D. and West, B. [1992]: *Analyzer* calculus software* (for the Macintosh computer), Reading, MA: Addison-Wesley.

EDUCOM/NCRIPTAL distinguished software for 1990 – for calculus and iteration of functions of a single variable.

Artigue, M. [1989]: Une recherche d'ingénierie didactique sur l'enseignement des équations différentielles en premier cycle universitaire in *Équipe de Didactique des Mathématiques et de l'Informatique: Siminaire de Didactique des Mathimatiques et de l'Informatique*, Grenoble, Année 1988-1989. Université de Grenoble, LSD-IMAG, Institut Fourier.

A didactical reflection on the prevalence of the algebraic approach to differential equations in undergraduate courses and construction of a new course with emphasis on geometric, qualitative and numerical elements.

Artigue, M. and Gautheron, V. [1983]: *Systèmes Différentielles: Etude Graphique*, CEDEC, Paris.

A pioneer work that deserves to be a classic. Teaches qualitative methods in the study of two-dimensional systems of autonomous differential equations with many beautiful designs.

Aspetsberger, K. and Kutzler, B. [1988]: *Symbolic computation — A new chance for education* in F. Lovis and E.D. Tagg (eds) *Computers in Education*, 331-336, Amsterdam: North-Holland.

Considers a wide range of symbolic computation activities with examples including computer algebra, computational geometry, automatic reasoning and automatic programming.

Banchoff, T. et al (Eds.) [1988]: *Educational Computing in Mathematics, ECM87*, Amsterdam: North Holland.

Proceedings of an international congress with presentations about a number of mathematical domains (differential geometry, calculus, dynamical systems, geometry, etc.) and reports about various teaching experiments.

Bolter, J.D. [1984]: *Turing's Man: Western Culture in the Computer Age*, Chapel Hill, NC: University of North Carolina Press.

An innovative analysis of the philosophical impact of computers as "embodied mathematics," of nature as information, and of humans as "information processors." Bolter draws interesting parallels with Athenian man—limited by finite mathematics, yet creating private universes like the craftsman potter in Plato's *Timaeus*.

Bushaw, D. [1983]: *A two-year lower-division mathematics sequence* in *The Future of College Mathematics*, A. Ralston and G.S. Young (Eds.), pp 111-118. New York: Springer-Verlag.

An article which presents an outline of a two-year undergraduate sequence which integrates calculus and discrete mathematics. Twenty-four different modules are listed as well as the interrelations among them.

Clocksin, W.F. and C.S. Mellish [1981]: *Programming in Prolog*, Berlin: Springer-Verlag.

This is a textbook for teaching Prolog as a programming language. It briefly discusses the logical foundations of the language but the emphasis is on how useful programs can be written using the Prolog systems that are available.

Computers and Mathematics, A column (past editor: J. Barwise, current editor: K. Devlin) appearing regularly in the Notices of the American Mathematical Society.

Papers appearing in this column discuss all aspects of the influence of computers on mathematics, both with respect to research and to teaching. Many Symbolic Mathematical Systems have been discussed — see Appendix 1 of the paper by B.R. Hodgson and E.R. Muller in this volume.

Cornu, B. [1992]: *L'Ordinateur pour enseigner les mathématiques*, Paris: Presses Universitaires de France.

A multi-author book with chapters on mathematics and informatics, on various examples of teaching mathematical concepts, and on the link between technology and educational research.

Devaney, R.L. [1990]: *Chaos, Fractals, and Dynamics: Computer Experiments in Mathematics*, Reading, MA: Addison-Wesley.

Paperback handbook full of explorations for either a classroom setting or individual study.

Douglas, R. (Ed.) [1986]: *Toward a Lean and Lively Calculus*, Proceedings of a Conference/Workshop at Tulane University, Washington, DC: Mathematical Association of America.

The report of a conference - sometimes called the counterreformation (in relation to discrete mathematics) - in which various recipes for improving the teaching of calculus are given.

Dubinsky, E. and Fraser, R. [1990]: *Computers and the Teaching of Mathematics*, Nottingham, UK: The Shell Centre for Mathematical Education.

This selection of papers from the Sixth International Congress on Mathematics Education contains those papers from ICME6 particularly relevant to the use of technology in teaching mathematics.

Dubinsky, E. and Schwingendorf, K. E. [1991]: *Constructing calculus concepts: cooperation in a computer laboratory* in *The Laboratory Approach to Teaching Calculus*, (Leinbach, L.C., Ed.), MAA Notes Series, Washington, DC: Mathematical Association of America.

Describes an early version of the Purdue University calculus project which emphasizes cooperative learning. Contains examples of computer assignments and examinations as well as details of the computer laboratory setup.

Fey, J.T. [1989]: Technology and Mathematics Education: A Survey of Recent Developments and Important Problems, *Educ. Studies in Math.*, 20, 237-272.

Provides an overview and analysis of recent progress in applying electronic information technology to the creation of new environments for intellectual work in mathematics. It discusses numerical computation, graphic computation and symbolic computation and contains many concrete examples.

Foster, K.R. and Bau, H. H. [1989]: Symbolic Manipulation Programs for the Personal Computer, *Science*, 243, 679-684.

Provides a useful comparative summary of the capabilities of a majority of Symbolic Mathematical Systems. The article provides prices and types of hardware required. (These systems are constantly changing and up to date information should be obtained from the manufacturers whose addresses are listed in the article.)

Garey, M.R. and D.S. Johnson [1978]: *Computers and Intractability, A Guide to the Theory of NP-Completeness*, San Francisco: Freeman.

This book includes a thorough introduction to complexity theory and $\mathcal{NP}$-complete problems. It shows how to recognize these problems and how to deal with them. It is a readable guide and offers an exhaustive list of $\mathcal{NP}$-complete problems.

Goldenberg, E.P. [1988]: Mathematics, Metaphors, and Human Factors: Mathematical, Technical, and Pedagogical Challenges in the Educational Use of Graphical Representation of Functions, *Journal of Mathematical Behavior*, 7, 2, 135-173.

Describes the work of a research group at the Educational Technology Center (ETC) at the Harvard Graduate School of Education. Many convincing examples show possible pitfalls and misunderstandings in the interpretation of function graphs.

Gray, T.W. and Glynn, J. [1991]: *Exploring Mathematics with Mathematica*, Reading, MA: Addison-Wesley.

An introduction to Mathematica, written in the form of a dialog between the two authors, presenting exploration of various mathematical concepts. A section is devoted to Mathematica's application to high school, college and university mathematics teaching. The book comes with a CD-ROM disk containing an electronic edition of the text in the form of Mathematica's Notebooks.

Heid, M.K., Sheets C. and Matras, M.A. [1990]: *Computer-enhanced Algebra: New roles and challenges for teachers and students* in T.J. Cooney and C.R. Hirsch (eds.) *Teaching and Learning Mathematics in the 1990s*, (1990 NCTM Yearbook), 194-204, Reston, VA: National Council of Teachers of Mathematics.

Discusses the new roles for teachers and students when the computer enters actively into mathematics education. The teacher as technical assistant. The teacher as collaborator. The teacher as facilitator and catalyst . Responsibilities for evaluating student learning, etc.

Hirst, A. and Hirst, K. (Eds.) [1988]: *Proceedings of the Sixth International Congress on Mathematics Education, Budapest, 1988*, Budapest: Janos Bolyai Mathematical Society.

The 'state-of-the-art' on the use of computers in mathematics education. Contains reports of plenary sessions on computerization of schools and mathematics education, and on algoritmic mathematics, and of a theme group on computers and the teaching of mathematics.

Hubbard, J.H. and West, B. [1991]: *MacMath: A Dynamics Package* (for the Macintosh computer), New York: Springer-Verlag.

Twelve interactive and easy-to-use graphics programs for both iteration and differential equations, with a handbook of suggestions for what to do with them; developed to accompany the authors' three volume text *Differential Equations: A Dynamical Systems Approach*also being published by Springer-Verlag.

Jacobsen, E. [1989]: *An International Perspective* in *SIGCUE OUTLOOK* (Bulletin of the Special Interest Group for Computer Uses in Education), 20, 2, New York: ACM Press.

Provides a worldwide overview of microcomputers in education. The priorities of governments and current usages are illustrated by selecting representative countries in each continent. UNESCO's role in the field of computers and education is examined.

Jaffe, A. [1984]: *Ordering the Universe: The Role of Mathematics* in *Renewing U.S. Mathematics*, Washington, DC: National Academy Press.

A sweeping survey of contemporary mathematics, "an ancient art, ... highly esoteric, and the most intensely practical of human endeavors." Emphasizes the role of mathematics in advancing computation, communication, physics, and engineering.

Johnson, D.C. and Lovis, F. (Eds.) [1987]: *Informatics and the Teaching of Mathematics*, Proceedings of the IFIP TC 3/WG 3.1 Working Conference on Informatics and the Teaching of Mathematics, Sofia, Bulgaria, 16 - 18 May, 1987. Amsterdam: North-Holland. ISBN 0-444-70325-X.

Many international contributions on a wide scope of themes, from theoretical to concrete, from primary school to college level.

Kenney, M., (Ed.), [1991]: *Discrete Mathematics across the Curriculum, K-12* (1991 NCTM Yearbook), Reston, VA: National Council of Teachers of Mathematics.

Discrete mathematics is the part of mathematics where algorithmics are most natural. This yearbook contains articles are specific implementation ideas at various levels.

Klotz, E. and Jackiw, N. [1991]: *The Geometer's Sketchpad* (for the Macintosh computer), Berkeley, CA: Key Curriculum Press.

Developed at Swarthmore College as part of the Visual Geometry Project under the direction of E. Klotz and D. Schattschneider. A dynamic tool for exploring geometry (e.g., you can construct a figure, distort it, and your construction follows the distortion.)

Knuth, D.E. [1973]: *The Art of Computer Programming: vol 3: Sorting and Searching*, Reading, MA: Addison-Wesley.

This is the third of Knuth's classical books. As with the other two, it should be considered as a source for information and references and not as a basic textbook. Although the title may sound as if the book is mainly for programmers concerned with preparing sorting routines, it virtually covers all theoretical aspects of programming.

Koçak, H. [1986]: *Phaser: Differential and Difference Equations through Computer Experiments* (for IBM computers), New York: Springer-Verlag.

The classic entry into the field of experimenting with differential equations.

Koerner, J.D. (Ed.) [1981]: *The New Liberal Arts: An Exchange of Views*, New York: Alfred P. Sloan Foundation.

A position paper followed by ten responses in which it is argued that analytic skills (e.g., statistics, computation, applied mathematics) are as crucial for liberal education as are traditional literary, historical, and artistic studies. The computer "has altered the world in which the student will live as well as the manner in which he will think about the world."

Krivine, J.L. and M. Parigot [1990]: Programming with proofs, *J. Inf. Process. Cybern.* EIK 26, 149-167.

This is one of the recent articles which develop the idea that proofs and programs are basically the same object. Even though it is mainly written for readers with a strong background in logic and computer science, it has deep insight on the subject.

LSW [1990]: Landesinstitut für Schule und Weiterbildung (LSW), Soest (Hrsg.): Neue Medien im Unterricht - Funktionenplotter im Mathematikunterricht. Soest: Soester Verlagskontor. ISBN 3-8165-1732-3.

This issue on the use of function plotters in mathematics teaching at schools gives general considerations on the didactical criteria, possibilities and limits of such tools, some concrete examples for the use at various places in the mathematical curriculum, and critical descriptions of some function plotters available in Germany.

Malkevitch, J. et al., [1988]: *For All Practical Purpose*, San Francisco: W. H. Freeman. Also available as 26 videotaped TV programs.

An innovative "general studies" secondary-tertiary course which shows students many current applications of elementary mathematics and shows faculty how many new topics (often algorithmic) can be included in the curriculum.

Mathematical Sciences Education Board [1990]: *Reshaping School Mathematics: A Philosophy and Framework for Curriculum*, Washington, DC: National Academy Press.

A detailed rationale for changing school mathematics, building on research related to the role of technology and to the process of teaching and learning. Summarizes (with extensive references) the relevant research literature; poses open questions; and outlines goals for curriculum reform.

Maurer, S. [1984]: Two meanings of algorithmic mathematics, *Math. Teacher*, 77, 430–435.

Explains at length the difference between the traditional and contemporary meanings of algorithmic mathematics. The two main examples are polynomial evaluation (mentioned briefly in Maurer's article here) and Gaussian elimination for solving systems of linear equations.

Maurer, S. [1985] The algorithmic way of life is best, *College Math. J.*, 16, 2–18 (Forum article and reply to responses).

The author presents a deliberately forceful argument for abandoning the traditional theory/computation schism in favor of an algorithmics synthesis. The article is followed by numerous thoughtful responses, some quite critical, and then a summary reply by the author.

Maurer, S. B. and Ralston, A. [1991]: *Discrete Algorithmic Mathematics*, Reading, MA: Addison-Wesley.

One of the most recent discrete mathematics texts. It is intended for well-prepared first-year university students and stresses the algorithmic approach to discrete mathematics.

Mines, B., Richman, F. and Ruitenberg, W. [1988]: *A Course in Constructive Algebra*, New York: Springer-Verlag.

An example of what classical pure mathematics may look like if researchers take the algorithmic viewpoint to heart. Topics such as factorization in polynomial rings are treated by defining algorithms and proving them correct. The review by Beeson (see references in Maurer, this volume) gives a thorough overview.

Muller, E.R. [1991]: *Maple Laboratory in a Service Calculus Course* in L.C. Leinbach et al (eds.) *The Laboratory Approach to Teaching Calculus*, MAA Notes Number 20, 111-117, Washington, DC: Mathematical Association of America.

Presents the development and implementation of compulsory laboratories. Includes examples of laboratory activities and provides data on traditional indicators (failure rates, etc.) and student attitudes.

National Council of Teachers of Mathematics [1989]: *Curriculum and Evaluation Standards for School Mathematics*, Reston, VA: National Council of Teachers of Mathematics.

A key document in America's attempt to come from way behind in mathematics education. Contains a detailed set of standards for school mathematics, arranged in four groups (K-4, 5-8, 9-12), giving expectations and examples in each curricular area. Builds on assumption of educating students for an information society; advocates extensive use of calculators and computers throughout school mathematics.

National Research Council [1989]: *Everybody Counts: A Report to the Nation on the Future of Mathematics Education*, Washington, DC: National Academy Press.

A call for action issued by the National Academy of Sciences to improve mathematics education in the United States. Highlights human resource needs, learning through involvement, and curriculum priorities. Stresses, among other things, the way computers have changed priorities for mathematics education.

Nievergelt, Y. [1987]: The Chip with the College Education: the HP-28C, *Amer. Math. Monthly*, 94, 895-902.

Details the power of the HP-28C by providing examples of its capabilities. Concludes that it "introduces one new element into the teaching of mathematics, namely awsome computing power at both modest price and size". (Which is even more true of the more recent HP-48SX.)

Okamori, H. [1989]: *Mathematics Education and Personal Computers*, Tokyo: Daiichi-Hoki Shuppan.

This volume is an excellent survey of computer use in mathematics education in Japan from kindergarten to university. It covers research and practice and includes a number of examples of problem solving in the real world (e.g. mathematics of a lake, road mathematics).

Page, W. [1990]: Computer Algebra Systems: Issues and Inquiries, *Computers Math. Applic.*, 19, 51-69.

An educational-philosophical survey article on issues of special importance to all who are involved with the instructional uses of computers in the mathematical sciences.

Peressini, A. et al [1992]: *Precalculus and Discrete Mathematics*, University of Chicago School Mathematics Project, Glenview, IL: Scott, Foresman.

The Chicago project has developed a highly innovative mathematics curriculum for average students in grades 7–12. This book is the 12th year text. It includes algorithmics. There are several other innovative projects under way in America which will also result in texts.

Ralston, A. [1981]: Computer Science, Mathematics, and the Undergraduate Curricula in Both, *Amer. Math. Monthly*, 88, 472-485.

An urgent appeal to mathematicians to recognize the fundamental mathematical requirements of computer science by giving discrete mathematics greater priority in early years of mathematical preparation. The beginning of a decade-long effort to establish discrete mathematics on an equal footing with calculus as an important foundation not only for computing but also for mathematics itself.

Ralston, A. and Young, G.S. (Eds.) [1983]: *The Future of College Mathematics*, New York: Springer-Verlag.

A report of the first conference at which discrete mathematics as a possible alternative to or coequal with calculus was considered. The papers discuss a wide variety of the relevant issues.

Rice, J.R. [1988]: Mathematical Aspects of Scientific Software in J.R. Rice (Ed.): *Mathematical Aspects of Scientific Software*, New York: Springer-Verlag and in The IMA Volumes in *Mathematics and Its Applications*, 14, 1 - 39.

Fundamental but concrete aspects of mathematics, applications and the new role these are taking when the users rely on ready-made mathematical methods.

Robinson, J.A. [1965]: A machine-oriented logic based on the resolution principle, *J. ACM* 12, 23-41.

This book gives an account of the impressive breakthrough achieved by its author towards performing deductive reasoning by a machine. It includes the presentation of the formalism of predicate logic, a thorough exposition of the resolution principle as well as a detailed account of a working computer program for showing "what follows from what".

Schmidt, Günter (Ed.) [1988]: ***Computer im Mathematikunterricht***, Der Mathematikunterricht 34, Heft 4. ISBN 3-617-24022-4, 19-42.

A special issue of a German journal aimed mainly at high school teachers. This issue contains three articles discussing the impact of computers on mathematics learning, analysing software for mathematics teaching, and describing a possible use of recursion in teaching calculus.

Steen, L.A. (Ed.) [1988]: ***Calculus for a New Century***, MAA Notes No. 8, Washington, DC: Mathematical Association of America.

A report of a conference whose papers look at the role of calculus and the teaching of calculus as we approach the 21st century.

Stern, J. [1990]: *Fondements Mathématiques de l'Informatique*, Paris: McGraw-Hill.

This is an undergraduate textbook which covers computability, complexity, logic and the theory of regular and algebraic sets. It is a readable introduction to the main tools and concepts of theoretical computer science.

Tall, D.O. [1986]: ***Building and Testing a Cognitive Approach to the Calculus Using Interactive Computer Graphics***, Ph.D. Thesis in Mathematics Education, The University of Warwick, Faculty of Education.

Combining mathematical, psychological and epistemological studies with the development of suitable software, important insights into the nature of the learning of the calculus are gained. The work is not application oriented, but the attempt to build up a true understanding using discrete and continuous aspects serves the user of mathematics, too.

Tall, D.O. [1986, 1990]: *Graphic Calculus I-III* (for BBC compatible computers), London: Glentop Press and, with P. Blokland and D. Kok, *A Geometric Approach to the Calculus* (for I.B.M. compatible computers), Sunburst, Inc., USA.

Interactive, extremely well designed, and easy-to-use graphics programs for all sorts of calculus topics, including multivariable, with excellent manuals for students and teachers.

Tall, D.O. [1987]: ***Readings in Mathematical Education: Understanding the Calculus***, collected articles from Mathematics Teaching, 1985-1987, Association of Teachers of Mathematics, UK.

These are among the earliest writings on the subject worldwide; they include many excellent insights and suggestions.

Tinsley, J.D. and van Weert, T.J. (Eds.) [1989]: ***Educational Software at the Secondary Level***, Amsterdam: Elsevier.

Proceedings of a 1989 IFIP working conference with many examples of educational software and discussions of the trends of software development and evolution.

Wagon, S. [1991]: *Mathematica in Action*, San Francisco: Freeman.

An example-based introduction to techniques, both elementary and advanced, of using Mathematica for mathematical computation and exploration. "An underlying theme of this book is that a computational way of looking at a mathematical problem or result yields many benefits."

West, B.J. [1985]: *An Essay on the Importance of Being Nonlinear*, Lecture Notes in Biomathematics 62, Berlin: Springer-Verlag. ISBN 3-540-16038-8.

Fundamental aspects of nonlinearity, mostly in the context of dynamical systems. A bit technical in some parts.

Wilf, H.S. [1982]: The Disk with the College Education, *Amer. Math. Monthly*, 89, 4-8.

An early effort to alert mathematicians to the power of symbolic computing systems—which are now much more powerful than those of a decade ago—and to the threat they pose for those who might continue the status quo in teaching undergraduate mathematics.

Zimmerman, W. and Cunningham, D. [1990]: *Visualization in Teaching and Learning Mathematics*, MAA Notes Number 19, Washington,DC: Mathematical Association of America.

An authoritative collection from many experts; topics include geometry, calculus, differential equations, differential geometry, complex analysis, linear algebra, iteration and stochastic processes.

Zorn, P. [1987]: Computing in Undergraduate Mathematics, *Notices of the American Mathematical Society*, 34, 917-923 (October).

A careful analysis of issues—philosophical, pedagogical, practical—associated with the introduction of computers and computer labs into undergraduate mathematics courses. Based on issues raised at a workshop of project leaders who are seeking to make these changes on different campuses.

Index

Abstraction, 35
Algorithm, 16, 20, 39f, 52f, 70-72, 73
    proof by, 17
Algorithm analysis, 46-47
Algorithm design, 45
Algorithm verification, 45
Algorithmic language, 49
Algorithmics, 45f
Analysis of Algorithms, 46-47
Approximation, 84
Arabic multiplication, 39
Artificial intelligence, 28
Assessment, 6
AUTOCALC, 63-64

Black boxes, 72-73
Blancmange function, 118
Bottom-up design, 45
Bourbaki, 3, 33

Calculator exercise, 44
Calculators for young children, 59
Calculus, 22f, 34, 80
    computers in, 23
CAYLEY, 13
Children, calculators for, 59
    computers for, 59
Classroom, computers in, 23, 35-36
Clique problem, 54
Complexity theory, 52
Computability, 51
Computation theory, 51
Computer-aided design, 58
Computer-aided instruction, 58
Computer-aided learning, 110
Computer algebra systems,
    see Symbolic mathematical systems
Computer-assisted learning, 29
Computer graphics, 23, 28, 117f
Computer literacy, 34-35
Computer science, 34, 37
    theoretical, 49, 50
Computers
    experimental use of, 112-114
    for mathematics teaching, 25f
    in the classroom, 23
    instruction with, 110
Continuous functions, 82
Courseware, 29

CUPM, 34
Curriculum, mathematics, 75-76
        in Japan, 76-77
        in the USA, 77
    primary, 59f
    secondary, 65f
    university, 20f, 82f
Curriculum change, 99-100

Data analysis, exploratory, 15
    software for, 70-71
Data structures, 53
Databases, 28
Difference equations, 49
Differential equations, 74, 118
    systems of, 120-121
Discovery in mathematics, 23
Discovery learning, 5
Discrete-continuous interplay, 110f
Discrete functions, 82
Discrete mathematics, 21, 34, 49, 80, 81f
Dynabook, 3
Dynamical systems, 112

Early science software, 60-62
Equations, numerical solution of, 43
Errors, 82
    student, 29
Estimation, 82
Euclid's algorithm, 40, 45, 47
Euler's method, 113
Experimentation in mathematics, 14, 65-66
Exploration in mathematics, 23
Exploratory data analysis, 15

Fast Fourier transform, 53
Four colour theorem, 14
Fractal, 16
Function, 12, 82
    blancmange, 118
    discrete, 82
    non-differentiable, 118
    plotting, 74
    recursive, 52
    software, 69-70
    Takagi, 13

Geometry, software for, 66-67
    visual exploration in, 121

Graph algorithms, 53
Graph theory, 49
Guessing, 36

Hamiltonian path problem, 53
Heuristics, 47
Hypermedia, 28

Induction, 46, 49
Instruction, individualized, 28
Integration, 82
Iterative methods, 15

Jacobi symbol, 53
Journey in Mathematics project, 7
Julia set, 16

KALEIDOSCOPE, 67-68
Key handling programs, 62
Kleene's theorem, 52

Language development software, 60
Laptop computers, 36
Learning, new styles of, 122
Levels of research and development, 9
Logic, 20, 54f
Loop invariant, 45

MacMath, 118
Mathematician, student as, 2
Mathematical communication, 17
Mathematical induction, 46, 49
Mathematical logic, 20, 54f
Mathematical metaphors, 36-37
Mathematical modelling, 15
Mathematics, discovery in, 23
    experimentation in, 14, 65-66
    exploration in, 23
    new approaches to, 117-118
    science of, 19
    university, 19
Mathematics programs, 63
Mathematics teaching, computers for, 25f
Matrix multiplication, 41
Model building, 74-76
Models, 111-112
Moving target problem, 1
Multimedia, 28

New mathematics, 3
    curricula for, 6, 31
        in Japan, 76-77
        in the USA, 77
Newton's Laws, 120
Non-differentiable functions, 118
NP-complete problems, 53
Numerical analysis, 20
Numerical solution of equations, 43

Operations research, 49

Partial correctness, 54
Pattern matching, 53
Plotting functions, 74
Pocket calculator, 29
Polynomial evaluation, 44, 46
Primality tests, 53
Primary schools:
    curriculum, 59f
    software for, 60f
Problem solving, 5
Program correctness, 54
Programming, 122
Prolog, 54
Proof, 13
    by algorithm, 17

Quadratic formula, 42-43

Random numbers, 43
Recursion theory, 52
Recursive function, 52
Representation, 84
Resolution, 54
Runge-Kutta method, 114

Satisfiability problem, 53
Searching, 49, 53
Secondary school curriculum, 65f
Self-evaluation, 28
Simplex method, 53
Simulation, 15, 73-74
    stochastic, 74
Smalltalk, 3
Software, 29f
    data analysis, 70-71
    for primary schools, 60f
    for secondary schools, 66f

function, 69-70
Soliton, 13
Sorting, 49, 53
Spreadsheet, 28
Square-root construction, 41-42, 47
Stochastic simulation, 74
Student errors, 29
Student-teacher relation, 26
Symbolic mathematical systems, 17, 72, 93f
  classroom sessions with, 97
  curriculum implications of, 99
  examples of use, 94-95, 105-106
  individual access to, 98
  laboratory sessions with, 97
  software reviews of, 104-105
Symbolic solutions, 112
Syntax analysis, 55
SYMMETRIC TURTLES, 66-67

Takagi function, 13
Teacher, role of, 27
Teacher-student relation, 26
Teacher training, 89f
Teaching, evolution of, 87
Teaching assistant, macro as, 7
Teaching-learning process, 109
Theoretical computer science, 49, 50
Top-down design, 45
Total correctness, 55
Towers of Hanoi, 42, 45, 46, 49
Training methodology, 90-91
Traveling salesman problem, 54
Turing machine, 52, 53

Unification-resolution, 54
University mathematics, curriculum for, 20
  preparation for, 19

Verification, algorithm, 45
Visual data processing, 121
Visual exploration, 121

White box, 73

Young children, calculators for, 59
  computers for, 59